libstdc++

Generated by Doxygen 1.16.1

1 Deprecated List 1

2 Todo List 2

3 Topic Documentation 4

3.1 Algorithms . . . 4
3.1.1 Detailed Description . . . 4
3.1.2 Generalized Numeric operations . . . 5
3.1.3 Mutating . . . 21
3.1.4 Non-Mutating . . . 43
3.1.5 Sorting . . . 62
3.2 Atomics . . . 104
3.2.1 Detailed Description . . . 109
3.2.2 Macro Definition Documentation . . . 109
3.2.3 Typedef Documentation . . . 109
3.2.4 Enumeration Type Documentation . . . 116
3.2.5 Function Documentation . . . 116
3.3 Concurrency . . . 116
3.3.1 Detailed Description . . . 117
3.3.2 Condition Variables . . . 117
3.3.3 Futures . . . 118
3.3.4 Mutexes . . . 121
3.3.5 Threads . . . 124
3.4 Containers . . . 125
3.4.1 Detailed Description . . . 126
3.4.2 Associative . . . 126
3.4.3 Sequences . . . 127
3.4.4 Unordered Associative . . . 128
3.5 Diagnostics . . . 129
3.5.1 Detailed Description . . . 130
3.5.2 Function Documentation . . . 130
3.5.3 Exceptions . . . 133
3.6 Extensions . . . 138
3.6.1 Detailed Description . . . 138
3.6.2 Dynamic Bitset. . . . 139
3.6.3 Policy-Based Data Structures . . . 143
3.6.4 SGI . . . 152
3.7 Filesystem . . . 161
3.7.1 Detailed Description . . . 166
3.7.2 Typedef Documentation . . . 166

3.7.3 Enumeration Type Documentation . . . 166
3.7.4 Function Documentation . . . 166
3.8 I/O . . . 167
3.8.1 Detailed Description . . . 169
3.8.2 Typedef Documentation . . . 169
3.9 Iterators . . . 171
3.9.1 Detailed Description . . . 175
3.9.2 Function Documentation . . . 175
3.9.3 Iterator Tags . . . 177
3.10 Locales . . . 177
3.10.1 Detailed Description . . . 178
3.10.2 Function Documentation . . . 178
3.11 Numerics . . . 180
3.11.1 Detailed Description . . . 180
3.11.2 Bit manipulation . . . 181
3.11.3 Complex Numbers . . . 181
3.11.4 Decimal Floating-Point Arithmetic . . . 190
3.11.5 Mathematical Special Functions . . . 191
3.11.6 Numeric Arrays . . . 220
3.11.7 Random Number Generation . . . 236
3.11.8 TR1 Mathematical Special Functions . . . 250
3.12 Ranges . . . 255
3.12.1 Detailed Description . . . 255
3.13 Regular Expressions . . . 256
3.13.1 Detailed Description . . . 259
3.13.2 Typedef Documentation . . . 259
3.13.3 Function Documentation . . . 260
3.13.4 Base and Implementation Classes . . . 277
3.14 Strings . . . 278
3.14.1 Detailed Description . . . 278
3.14.2 Typedef Documentation . . . 278
3.15 Technical Specifications . . . 279
3.15.1 Detailed Description . . . 279
3.15.2 Filesystem TS . . . 280
3.15.3 Library Fundamentals TS . . . 285
3.15.4 Parallelism TS . . . 300
3.16 Utilities . . . 301
3.16.1 Detailed Description . . . 304
3.16.2 Function Documentation . . . 304

3.16.3 Variable Documentation . . . 313
3.16.4 Concepts . . . 314
3.16.5 Function Objects . . . 315
3.16.6 Memory . . . 325
3.16.7 Metaprogramming . . . 354
3.16.8 Rational Arithmetic . . . 363
3.16.9 Time . . . 365

4 Namespace Documentation **386**

4.1 __gnu_cxx Namespace Reference . . . 386
4.1.1 Detailed Description . . . 403
4.1.2 Typedef Documentation . . . 403
4.1.3 Function Documentation . . . 403
4.2 __gnu_cxx::__detail Namespace Reference . . . 413
4.2.1 Detailed Description . . . 413
4.2.2 Function Documentation . . . 414
4.3 __gnu_cxx::typelist Namespace Reference . . . 414
4.3.1 Detailed Description . . . 414
4.3.2 Function Documentation . . . 415
4.4 __gnu_debug Namespace Reference . . . 415
4.4.1 Detailed Description . . . 420
4.4.2 Typedef Documentation . . . 421
4.4.3 Enumeration Type Documentation . . . 421
4.4.4 Function Documentation . . . 421
4.5 __gnu_internal Namespace Reference . . . 423
4.5.1 Detailed Description . . . 423
4.6 __gnu_parallel Namespace Reference . . . 423
4.6.1 Detailed Description . . . 430
4.6.2 Typedef Documentation . . . 431
4.6.3 Enumeration Type Documentation . . . 431
4.6.4 Function Documentation . . . 432
4.6.5 Variable Documentation . . . 472
4.7 __gnu_pbds Namespace Reference . . . 473
4.7.1 Detailed Description . . . 475
4.8 __gnu_sequential Namespace Reference . . . 475
4.8.1 Detailed Description . . . 475
4.9 abi Namespace Reference . . . 475
4.9.1 Detailed Description . . . 475
4.10 std Namespace Reference . . . 475

4.10.1 Detailed Description . . . 587
4.10.2 Typedef Documentation . . . 587
4.10.3 Enumeration Type Documentation . . . 589
4.10.4 Function Documentation . . . 590
4.10.5 Variable Documentation . . . 649
4.11 std::__debug Namespace Reference . . . 650
4.11.1 Detailed Description . . . 657
4.11.2 Function Documentation . . . 657
4.12 std::__detail Namespace Reference . . . 657
4.12.1 Detailed Description . . . 660
4.12.2 Function Documentation . . . 660
4.13 std::__parallel Namespace Reference . . . 661
4.13.1 Detailed Description . . . 678
4.13.2 Function Documentation . . . 678
4.14 std::chrono Namespace Reference . . . 679
4.14.1 Detailed Description . . . 686
4.15 std::decimal Namespace Reference . . . 686
4.15.1 Detailed Description . . . 695
4.15.2 Function Documentation . . . 695
4.16 std::experimental Namespace Reference . . . 695
4.16.1 Detailed Description . . . 706
4.16.2 Function Documentation . . . 706
4.16.3 Variable Documentation . . . 707
4.17 std::filesystem Namespace Reference . . . 708
4.17.1 Detailed Description . . . 711
4.18 std::literals Namespace Reference . . . 711
4.18.1 Detailed Description . . . 712
4.19 std::literals::chrono_literals Namespace Reference . . . 712
4.19.1 Detailed Description . . . 712
4.20 std::placeholders Namespace Reference . . . 713
4.20.1 Detailed Description . . . 713
4.21 std::regex_constants Namespace Reference . . . 713
4.21.1 Detailed Description . . . 715
4.21.2 Enumeration Type Documentation . . . 715
4.21.3 Function Documentation . . . 715
4.21.4 Variable Documentation . . . 719
4.22 std::rel_ops Namespace Reference . . . 723
4.22.1 Detailed Description . . . 723
4.22.2 Function Documentation . . . 723

4.23 std::this_thread Namespace Reference . . . 725
4.23.1 Detailed Description . . . 725
4.23.2 Function Documentation . . . 725
4.24 std::tr1 Namespace Reference . . . 726
4.24.1 Detailed Description . . . 729
4.24.2 Function Documentation . . . 729
4.25 std::tr1::__detail Namespace Reference . . . 731
4.25.1 Detailed Description . . . 731
4.26 std::tr2 Namespace Reference . . . 731
4.26.1 Detailed Description . . . 733
4.27 std::tr2::__detail Namespace Reference . . . 733
4.27.1 Detailed Description . . . 733

5 Concept Documentation **733**

5.1 std::assignable_from Concept Reference . . . 733
5.1.1 Concept definition . . . 733
5.1.2 Detailed Description . . . 733
5.2 std::common_reference_with Concept Reference . . . 733
5.2.1 Concept definition . . . 734
5.2.2 Detailed Description . . . 734
5.3 std::common_with Concept Reference . . . 734
5.3.1 Concept definition . . . 734
5.3.2 Detailed Description . . . 734
5.4 std::constructible_from Concept Reference . . . 734
5.4.1 Concept definition . . . 734
5.4.2 Detailed Description . . . 734
5.5 std::convertible_to Concept Reference . . . 734
5.5.1 Concept definition . . . 734
5.5.2 Detailed Description . . . 734
5.6 std::copy_constructible Concept Reference . . . 735
5.6.1 Concept definition . . . 735
5.6.2 Detailed Description . . . 735
5.7 std::default_initializable Concept Reference . . . 735
5.7.1 Concept definition . . . 735
5.7.2 Detailed Description . . . 735
5.8 std::derived_from Concept Reference . . . 735
5.8.1 Concept definition . . . 735
5.8.2 Detailed Description . . . 735
5.9 std::destructible Concept Reference . . . 735

5.9.1 Concept definition . . . 735
5.9.2 Detailed Description . . . 735
5.10 std::equivalence_relation Concept Reference . . . 736
5.10.1 Concept definition . . . 736
5.10.2 Detailed Description . . . 736
5.11 std::indirectly_comparable Concept Reference . . . 736
5.11.1 Concept definition . . . 736
5.11.2 Detailed Description . . . 736
5.12 std::indirectly_copyable Concept Reference . . . 736
5.12.1 Concept definition . . . 736
5.12.2 Detailed Description . . . 736
5.13 std::indirectly_movable Concept Reference . . . 736
5.13.1 Concept definition . . . 736
5.13.2 Detailed Description . . . 736
5.14 std::indirectly_readable Concept Reference . . . 736
5.14.1 Concept definition . . . 737
5.14.2 Detailed Description . . . 737
5.15 std::indirectly_swappable Concept Reference . . . 737
5.15.1 Concept definition . . . 737
5.15.2 Detailed Description . . . 737
5.16 std::indirectly_writable Concept Reference . . . 737
5.16.1 Concept definition . . . 737
5.16.2 Detailed Description . . . 737
5.17 std::invocable Concept Reference . . . 737
5.17.1 Concept definition . . . 737
5.17.2 Detailed Description . . . 737
5.18 std::mergeable Concept Reference . . . 738
5.18.1 Concept definition . . . 738
5.18.2 Detailed Description . . . 738
5.19 std::move_constructible Concept Reference . . . 738
5.19.1 Concept definition . . . 738
5.19.2 Detailed Description . . . 738
5.20 std::permutable Concept Reference . . . 738
5.20.1 Concept definition . . . 738
5.20.2 Detailed Description . . . 738
5.21 std::predicate Concept Reference . . . 738
5.21.1 Concept definition . . . 738
5.21.2 Detailed Description . . . 738
5.22 std::ranges::bidirectional_range Concept Reference . . . 739

5.22.1 Concept definition . . . 739
5.22.2 Detailed Description . . . 739
5.23 std::ranges::borrowed_range Concept Reference . . . 739
5.23.1 Concept definition . . . 739
5.23.2 Detailed Description . . . 739
5.24 std::ranges::common_range Concept Reference . . . 739
5.24.1 Concept definition . . . 739
5.24.2 Detailed Description . . . 739
5.25 std::ranges::contiguous_range Concept Reference . . . 739
5.25.1 Concept definition . . . 739
5.25.2 Detailed Description . . . 739
5.26 std::ranges::forward_range Concept Reference . . . 739
5.26.1 Concept definition . . . 740
5.26.2 Detailed Description . . . 740
5.27 std::ranges::input_range Concept Reference . . . 740
5.27.1 Concept definition . . . 740
5.27.2 Detailed Description . . . 740
5.28 std::ranges::output_range Concept Reference . . . 740
5.28.1 Concept definition . . . 740
5.28.2 Detailed Description . . . 740
5.29 std::ranges::random_access_range Concept Reference . . . 740
5.29.1 Concept definition . . . 740
5.29.2 Detailed Description . . . 740
5.30 std::ranges::range Concept Reference . . . 740
5.30.1 Concept definition . . . 741
5.30.2 Detailed Description . . . 741
5.31 std::ranges::sized_range Concept Reference . . . 741
5.31.1 Concept definition . . . 741
5.31.2 Detailed Description . . . 741
5.32 std::ranges::view Concept Reference . . . 741
5.32.1 Concept definition . . . 741
5.32.2 Detailed Description . . . 741
5.33 std::ranges::viewable_range Concept Reference . . . 741
5.33.1 Concept definition . . . 741
5.33.2 Detailed Description . . . 741
5.34 std::regular_invocable Concept Reference . . . 741
5.34.1 Concept definition . . . 742
5.34.2 Detailed Description . . . 742
5.35 std::relation Concept Reference . . . 742

5.35.1 Concept definition . . . 742
5.35.2 Detailed Description . . . 742
5.36 std::same_as Concept Reference . . . 742
5.36.1 Concept definition . . . 742
5.36.2 Detailed Description . . . 742
5.37 std::simd::__vec_builtin Concept Reference . . . 742
5.37.1 Concept definition . . . 742
5.37.2 Detailed Description . . . 742
5.38 std::simd::__vec_builtin_of Concept Reference . . . 742
5.38.1 Concept definition . . . 743
5.38.2 Detailed Description . . . 743
5.39 std::sortable Concept Reference . . . 743
5.39.1 Concept definition . . . 743
5.39.2 Detailed Description . . . 743
5.40 std::strict_weak_order Concept Reference . . . 743
5.40.1 Concept definition . . . 743
5.40.2 Detailed Description . . . 743
5.41 std::uniform_random_bit_generator Concept Reference . . . 743
5.41.1 Concept definition . . . 743
5.41.2 Detailed Description . . . 743
5.42 std::weakly_incrementable Concept Reference . . . 744
5.42.1 Concept definition . . . 744
5.42.2 Detailed Description . . . 744

6 Class Documentation **744**

6.1 __gnu_parallel::__accumulate_binop_reduct< _BinOp > Struct Template Reference . . . 744
6.1.1 Detailed Description . . . 744
6.2 __gnu_parallel::__accumulate_selector< _It > Struct Template Reference . . . 744
6.2.1 Detailed Description . . . 745
6.2.2 Member Function Documentation . . . 745
6.2.3 Member Data Documentation . . . 746
6.3 __gnu_parallel::__adjacent_difference_selector< _It > Struct Template Reference . . . 746
6.3.1 Detailed Description . . . 747
6.3.2 Member Data Documentation . . . 747
6.4 __gnu_parallel::__adjacent_find_selector Struct Reference . . . 747
6.4.1 Detailed Description . . . 747
6.4.2 Member Function Documentation . . . 748
6.5 __gnu_cxx::__alloc_traits< _Alloc, typename > Struct Template Reference . . . 748
6.5.1 Detailed Description . . . 750

6.5.2 Member Typedef Documentation . . . 750
6.5.3 Member Function Documentation . . . 751
6.6 std::__basic_future< _Res > Class Template Reference . . . 755
6.6.1 Detailed Description . . . 756
6.6.2 Member Function Documentation . . . 757
6.7 __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference . . . 757
6.7.1 Detailed Description . . . 758
6.7.2 Member Typedef Documentation . . . 758
6.8 __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference . . . 758
6.8.1 Detailed Description . . . 759
6.8.2 Member Typedef Documentation . . . 759
6.9 std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class Template Reference . . . 759
6.9.1 Detailed Description . . . 760
6.9.2 Member Function Documentation . . . 760
6.10 __gnu_cxx::__common_pool_policy< _PoolTp, _Thread > Struct Template Reference . . . 763
6.10.1 Detailed Description . . . 763
6.11 __gnu_parallel::__count_if_selector< _It, _Diff > Struct Template Reference . . . 763
6.11.1 Detailed Description . . . 764
6.11.2 Member Function Documentation . . . 764
6.11.3 Member Data Documentation . . . 765
6.12 __gnu_parallel::__count_selector< _It, _Diff > Struct Template Reference . . . 765
6.12.1 Detailed Description . . . 766
6.12.2 Member Function Documentation . . . 766
6.12.3 Member Data Documentation . . . 766
6.13 std::__ctype_abstract_base< _CharT > Class Template Reference . . . 766
6.13.1 Detailed Description . . . 768
6.13.2 Member Typedef Documentation . . . 768
6.13.3 Member Function Documentation . . . 769
6.14 std::tr2::__dynamic_bitset_base< _WordT, _Alloc > Struct Template Reference . . . 780
6.14.1 Detailed Description . . . 782
6.14.2 Member Data Documentation . . . 782
6.15 __gnu_parallel::__fill_selector< _It > Struct Template Reference . . . 782
6.15.1 Detailed Description . . . 782
6.15.2 Member Function Documentation . . . 783
6.15.3 Member Data Documentation . . . 783
6.16 __gnu_parallel::__find_first_of_selector< _FIterator > Struct Template Reference . . . 783
6.16.1 Detailed Description . . . 784
6.16.2 Member Function Documentation . . . 784

6.17 __gnu_parallel::__find_if_selector Struct Reference . . . 785
6.17.1 Detailed Description . . . 785
6.17.2 Member Function Documentation . . . 785
6.18 __gnu_parallel::__for_each_selector< _It > Struct Template Reference . . . 786
6.18.1 Detailed Description . . . 787
6.18.2 Member Function Documentation . . . 787
6.18.3 Member Data Documentation . . . 788
6.19 __cxxabiv1::__forced_unwind Class Reference . . . 788
6.19.1 Detailed Description . . . 788
6.20 __gnu_parallel::__generate_selector< _It > Struct Template Reference . . . 788
6.20.1 Detailed Description . . . 789
6.20.2 Member Function Documentation . . . 789
6.20.3 Member Data Documentation . . . 789
6.21 __gnu_parallel::__generic_find_selector Struct Reference . . . 789
6.21.1 Detailed Description . . . 790
6.22 __gnu_parallel::__generic_for_each_selector< _It > Struct Template Reference . . . 790
6.22.1 Detailed Description . . . 792
6.22.2 Member Data Documentation . . . 792
6.23 __gnu_parallel::__identity_selector< _It > Struct Template Reference . . . 792
6.23.1 Detailed Description . . . 792
6.23.2 Member Function Documentation . . . 793
6.23.3 Member Data Documentation . . . 793
6.24 __gnu_parallel::__inner_product_selector< _It, _It2, _Tp > Struct Template Reference . . . 793
6.24.1 Detailed Description . . . 794
6.24.2 Constructor & Destructor Documentation . . . 794
6.24.3 Member Function Documentation . . . 795
6.24.4 Member Data Documentation . . . 795
6.25 std::__is_fast_hash< _Hash > Struct Template Reference . . . 795
6.25.1 Detailed Description . . . 796
6.26 std::__is_location_invariant< _Tp > Struct Template Reference . . . 796
6.26.1 Detailed Description . . . 796
6.27 __gnu_parallel::__max_element_reduct< _Compare, _It > Struct Template Reference . . . 796
6.27.1 Detailed Description . . . 796
6.28 __gnu_parallel::__min_element_reduct< _Compare, _It > Struct Template Reference . . . 796
6.28.1 Detailed Description . . . 797
6.29 __gnu_cxx::__detail::__mini_vector< _Tp > Class Template Reference . . . 797
6.29.1 Detailed Description . . . 797
6.30 __gnu_parallel::__mismatch_selector Struct Reference . . . 798
6.30.1 Detailed Description . . . 798

6.30.2 Member Function Documentation 798
6.31 __gnu_cxx::__mt_alloc< _Tp, _Poolp > Class Template Reference 799
6.31.1 Detailed Description 800
6.32 __gnu_cxx::__mt_alloc_base< _Tp > Class Template Reference 800
6.32.1 Detailed Description 801
6.33 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference 801
6.33.1 Detailed Description 802
6.34 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference 802
6.34.1 Detailed Description 802
6.35 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference 802
6.35.1 Detailed Description 802
6.36 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference 803
6.36.1 Detailed Description 803
6.37 __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference 803
6.37.1 Detailed Description 803
6.38 __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference 803
6.38.1 Detailed Description 804
6.39 std::__new_allocator< _Tp > Class Template Reference 804
6.39.1 Detailed Description 805
6.40 std::__numeric_limits_base Struct Reference 805
6.40.1 Detailed Description 807
6.40.2 Member Data Documentation 807
6.41 __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread > Struct Template Reference 810
6.41.1 Detailed Description 810
6.42 __gnu_cxx::__pool< _Thread > Class Template Reference 810
6.42.1 Detailed Description 810
6.43 __gnu_cxx::__pool< false > Class Reference 810
6.43.1 Detailed Description 811
6.44 __gnu_cxx::__pool< true > Class Reference 811
6.44.1 Detailed Description 812
6.45 __gnu_cxx::__pool_alloc< _Tp > Class Template Reference 812
6.45.1 Detailed Description 813
6.46 __gnu_cxx::__pool_alloc_base Class Reference 813
6.46.1 Detailed Description 814
6.47 __gnu_cxx::__pool_base Struct Reference 814

6.47.1 Detailed Description . . . 815
6.48 __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc > Class Template Reference . . . 815
6.48.1 Detailed Description . . . 818
6.49 std::tr2::__reflection_typelist< _Elements > Struct Template Reference . . . 818
6.49.1 Detailed Description . . . 818
6.50 std::tr2::__reflection_typelist< _First, _Rest... > Struct Template Reference . . . 818
6.50.1 Detailed Description . . . 818
6.51 std::tr2::__reflection_typelist<> Struct Reference . . . 819
6.51.1 Detailed Description . . . 819
6.52 __gnu_parallel::__replace_if_selector< _It, _Op, _Tp > Struct Template Reference . . . 819
6.52.1 Detailed Description . . . 819
6.52.2 Constructor & Destructor Documentation . . . 820
6.52.3 Member Function Documentation . . . 820
6.52.4 Member Data Documentation . . . 820
6.53 __gnu_parallel::__replace_selector< _It, _Tp > Struct Template Reference . . . 821
6.53.1 Detailed Description . . . 821
6.53.2 Constructor & Destructor Documentation . . . 821
6.53.3 Member Function Documentation . . . 822
6.53.4 Member Data Documentation . . . 822
6.54 __gnu_cxx::__scoped_lock Class Reference . . . 822
6.54.1 Detailed Description . . . 822
6.55 __gnu_parallel::__transform1_selector< _It > Struct Template Reference . . . 823
6.55.1 Detailed Description . . . 823
6.55.2 Member Function Documentation . . . 823
6.55.3 Member Data Documentation . . . 824
6.56 __gnu_parallel::__transform2_selector< _It > Struct Template Reference . . . 824
6.56.1 Detailed Description . . . 825
6.56.2 Member Function Documentation . . . 825
6.56.3 Member Data Documentation . . . 825
6.57 __gnu_parallel::__unary_negate< _Predicate, argument_type > Class Template Reference . . . 825
6.57.1 Detailed Description . . . 826
6.57.2 Member Typedef Documentation . . . 826
6.58 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > Class Template Reference . . . 827
6.58.1 Detailed Description . . . 830
6.58.2 Constructor & Destructor Documentation . . . 830
6.58.3 Member Function Documentation . . . 833
6.58.4 Member Data Documentation . . . 879
6.59 __gnu_debug::_After_nth_from< _Iterator > Class Template Reference . . . 880
6.59.1 Detailed Description . . . 880

6.60 std::_Base_bitset< _Nw > Struct Template Reference 880
6.60.1 Detailed Description 881
6.60.2 Member Data Documentation 881
6.61 std::_Base_bitset< 0 > Struct Reference 881
6.61.1 Detailed Description 882
6.61.2 Member Data Documentation 882
6.62 std::_Base_bitset< 1 > Struct Reference 882
6.62.1 Detailed Description 883
6.63 __gnu_debug::_BeforeBeginHelper< _Sequence > Struct Template Reference 883
6.63.1 Detailed Description 884
6.64 std::_Bind< _Signature > Class Template Reference 884
6.64.1 Detailed Description 884
6.65 std::_Bind_result< _Result, _Signature > Class Template Reference 884
6.65.1 Detailed Description 884
6.66 __gnu_cxx::__detail::_Bitmap_counter< _Tp > Class Template Reference 884
6.66.1 Detailed Description 884
6.67 std::__detail::_BracketMatcher< _TraitsT, __icase, __collate > Struct Template Reference 885
6.67.1 Detailed Description 885
6.68 __gnu_cxx::_Caster< _ToType > Struct Template Reference 885
6.68.1 Detailed Description 885
6.69 __gnu_cxx::_Char_types< _CharT > Struct Template Reference 886
6.69.1 Detailed Description 886
6.70 __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > Class Template Reference 886
6.70.1 Detailed Description 888
6.71 std::__detail::_Compiler< _TraitsT > Class Template Reference 888
6.71.1 Detailed Description 889
6.72 std::__parallel::_CRandNumber< _MustBeInt > Struct Template Reference 889
6.72.1 Detailed Description 889
6.73 std::_Deque_base< _Tp, _Alloc > Class Template Reference 889
6.73.1 Detailed Description 890
6.73.2 Member Function Documentation 890
6.74 std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference 891
6.74.1 Detailed Description 892
6.74.2 Member Function Documentation 892
6.75 __gnu_parallel::_DRandomShufflingGlobalData< _RAIter > Struct Template Reference 892
6.75.1 Detailed Description 893
6.75.2 Constructor & Destructor Documentation 893
6.75.3 Member Data Documentation 893

6.76 __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator > Struct Template Reference . . 894
6.76.1 Detailed Description . . . 894
6.76.2 Member Data Documentation . . . 895
6.77 __gnu_parallel::_DummyReduct Struct Reference . . . 895
6.77.1 Detailed Description . . . 895
6.78 __gnu_debug::_Equal_to< _Type > Class Template Reference . . . 896
6.78.1 Detailed Description . . . 896
6.79 __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare > Class Template Reference . . . 896
6.79.1 Detailed Description . . . 897
6.79.2 Member Typedef Documentation . . . 897
6.80 __gnu_parallel::_EqualTo< _T1, _T2 > Struct Template Reference . . . 897
6.80.1 Detailed Description . . . 898
6.80.2 Member Typedef Documentation . . . 898
6.81 std::__detail::_Executor< _BiIter, _Alloc, _TraitsT > Class Template Reference . . . 898
6.81.1 Detailed Description . . . 899
6.82 __gnu_cxx::_ExtPtr_allocator< _Tp > Class Template Reference . . . 899
6.82.1 Detailed Description . . . 900
6.83 __gnu_cxx::__detail::_Ffit_finder< _Tp > Class Template Reference . . . 900
6.83.1 Detailed Description . . . 900
6.84 std::_Function_base Class Reference . . . 901
6.84.1 Detailed Description . . . 901
6.85 std::_Fwd_list_base< _Tp, _Alloc > Struct Template Reference . . . 901
6.85.1 Detailed Description . . . 903
6.86 std::_Fwd_list_const_iterator< _Tp > Struct Template Reference . . . 903
6.86.1 Detailed Description . . . 903
6.86.2 Friends And Related Symbol Documentation . . . 904
6.87 std::_Fwd_list_iterator< _Tp > Struct Template Reference . . . 904
6.87.1 Detailed Description . . . 904
6.87.2 Friends And Related Symbol Documentation . . . 905
6.88 std::_Fwd_list_node< _Tp > Struct Template Reference . . . 905
6.88.1 Detailed Description . . . 906
6.89 std::_Fwd_list_node_base Struct Reference . . . 906
6.89.1 Detailed Description . . . 907
6.90 __gnu_parallel::_GuardedIterator< _RAIter, _Compare > Class Template Reference . . . 907
6.90.1 Detailed Description . . . 907
6.90.2 Constructor & Destructor Documentation . . . 907
6.90.3 Member Function Documentation . . . 908
6.90.4 Friends And Related Symbol Documentation . . . 908
6.91 __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata > Struct Template Reference . . . 909

6.91.1 Detailed Description . . . 910
6.92 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata > Struct Template Reference . . . 910
6.92.1 Detailed Description . . . 911
6.93 __gnu_cxx::_Invalid_type Struct Reference . . . 912
6.93.1 Detailed Description . . . 912
6.94 __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator > Class Template Reference . . . 912
6.94.1 Detailed Description . . . 913
6.95 std::__fwdlist::_Iterator< _Const, _Ptr > Class Template Reference . . . 913
6.95.1 Detailed Description . . . 914
6.95.2 Friends And Related Symbol Documentation . . . 914
6.96 __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory > Class Template Reference . . . 914
6.96.1 Detailed Description . . . 917
6.96.2 Member Typedef Documentation . . . 917
6.96.3 Member Function Documentation . . . 917
6.96.4 Friends And Related Symbol Documentation . . . 919
6.96.5 Member Data Documentation . . . 920
6.97 __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory > Class Template Reference . . . 921
6.97.1 Detailed Description . . . 921
6.98 __gnu_parallel::_Job< _DifferenceTp > Struct Template Reference . . . 921
6.98.1 Detailed Description . . . 922
6.98.2 Member Data Documentation . . . 922
6.99 __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata > Struct Template Reference . . . 922
6.99.1 Detailed Description . . . 923
6.100 __gnu_parallel::_Less< _T1, _T2 > Struct Template Reference . . . 924
6.100.1 Detailed Description . . . 924
6.100.2 Member Typedef Documentation . . . 924
6.101 __gnu_parallel::_Lexicographic< _T1, _T2, _Compare > Class Template Reference . . . 925
6.101.1 Detailed Description . . . 925
6.101.2 Member Typedef Documentation . . . 925
6.102 __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare > Class Template Reference . . . 926
6.102.1 Detailed Description . . . 927
6.102.2 Member Typedef Documentation . . . 927
6.103 std::_List_base< _Tp, _Alloc > Class Template Reference . . . 927
6.103.1 Detailed Description . . . 928
6.104 std::_List_const_iterator< _Tp > Struct Template Reference . . . 929
6.104.1 Detailed Description . . . 929
6.105 std::_List_iterator< _Tp > Struct Template Reference . . . 929
6.105.1 Detailed Description . . . 930

6.106 std::_List_node< _Tp > Struct Template Reference 930
6.106.1 Detailed Description 931
6.107 std::__detail::_List_node_base Struct Reference 932
6.107.1 Detailed Description 932
6.108 std::__detail::_List_node_header Struct Reference 933
6.108.1 Detailed Description 933
6.109 __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser Struct Reference 934
6.109.1 Detailed Description 934
6.109.2 Member Data Documentation 934
6.110 __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser Struct Reference 934
6.110.1 Detailed Description 934
6.111 __gnu_parallel::_LoserTree< __stable, _Tp, _Compare > Class Template Reference 935
6.111.1 Detailed Description 935
6.111.2 Member Function Documentation 935
6.111.3 Member Data Documentation 936
6.112 __gnu_parallel::_LoserTree< false, _Tp, _Compare > Class Template Reference 936
6.112.1 Detailed Description 937
6.112.2 Member Function Documentation 937
6.112.3 Member Data Documentation 938
6.113 __gnu_parallel::_LoserTreeBase< _Tp, _Compare > Class Template Reference 939
6.113.1 Detailed Description 939
6.113.2 Constructor & Destructor Documentation 940
6.113.3 Member Function Documentation 940
6.113.4 Member Data Documentation 941
6.114 __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare > Class Template Reference 941
6.114.1 Detailed Description 942
6.115 __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare > Class Template Reference 942
6.115.1 Detailed Description 943
6.116 __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare > Class Template Reference 943
6.116.1 Detailed Description 944
6.117 __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare > Class Template Reference 944
6.117.1 Detailed Description 945
6.118 __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare > Class Template Reference . . 945
6.118.1 Detailed Description 946
6.119 __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare > Class Template Reference . . . 946
6.119.1 Detailed Description 947
6.120 __gnu_parallel::_LoserTreeTraits< _Tp > Struct Template Reference 947
6.120.1 Detailed Description 948
6.120.2 Member Data Documentation 948

6.121 __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare > Class Template Reference 948
6.121.1 Detailed Description 949
6.122 __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare > Class Template Reference 949
6.122.1 Detailed Description 950
6.123 __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare > Class Template Reference 950
6.123.1 Detailed Description 951
6.124 __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc > Struct Template Reference 951
6.124.1 Detailed Description 951
6.125 __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc > Struct Template Reference 951
6.125.1 Detailed Description 952
6.126 __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result > Struct Template Reference 952
6.126.1 Detailed Description 952
6.126.2 Member Typedef Documentation 953
6.127 std::__fwdlist::_Node< _ValPtr > Struct Template Reference 953
6.127.1 Detailed Description 954
6.128 __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata > Struct Template Reference . . . 954
6.128.1 Detailed Description 955
6.129 std::__fwdlist::_Node_base< _VoidPtr > Struct Template Reference 955
6.129.1 Detailed Description 955
6.130 __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference 956
6.130.1 Detailed Description 957
6.130.2 Member Typedef Documentation 957
6.130.3 Member Function Documentation 957
6.131 __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference 958
6.131.1 Detailed Description 960
6.131.2 Member Typedef Documentation 960
6.131.3 Member Function Documentation 960
6.132 __gnu_debug::_Not_equal_to< _Type > Class Template Reference 961
6.132.1 Detailed Description 961
6.133 std::_Not_fn< _Fn > Class Template Reference 962
6.133.1 Detailed Description 962
6.134 __gnu_parallel::_Nothing Struct Reference 962
6.134.1 Detailed Description 963
6.134.2 Member Function Documentation 963
6.135 __gnu_parallel::_Piece< _DifferenceTp > Struct Template Reference 963
6.135.1 Detailed Description 963
6.135.2 Member Data Documentation 963
6.136 std::_Placeholder< _Num > Struct Template Reference 964

6.136.1 Detailed Description . . . 964
6.137 __gnu_parallel::_Plus< _Tp1, _Tp2, _Result > Struct Template Reference . . . 964
6.137.1 Detailed Description . . . 965
6.137.2 Member Typedef Documentation . . . 965
6.138 __gnu_parallel::_PMWMSSortingData< _RAIter > Struct Template Reference . . . 965
6.138.1 Detailed Description . . . 966
6.138.2 Member Data Documentation . . . 966
6.139 __gnu_cxx::_Pointer_adapter< _Storage_policy > Class Template Reference . . . 967
6.139.1 Detailed Description . . . 968
6.140 __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp > Class Template Reference . . . 969
6.140.1 Detailed Description . . . 969
6.140.2 Constructor & Destructor Documentation . . . 969
6.140.3 Member Function Documentation . . . 970
6.141 __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp > Class Template Reference . . . 970
6.141.1 Detailed Description . . . 970
6.142 __gnu_parallel::_QSBThreadLocal< _RAIter > Struct Template Reference . . . 971
6.142.1 Detailed Description . . . 971
6.142.2 Member Typedef Documentation . . . 971
6.142.3 Constructor & Destructor Documentation . . . 971
6.142.4 Member Data Documentation . . . 972
6.143 std::__detail::_Quoted_string< _String, _CharT > Struct Template Reference . . . 972
6.143.1 Detailed Description . . . 973
6.144 __gnu_parallel::_RandomNumber Class Reference . . . 973
6.144.1 Detailed Description . . . 973
6.144.2 Constructor & Destructor Documentation . . . 973
6.144.3 Member Function Documentation . . . 974
6.145 __gnu_cxx::_Relative_pointer_impl< _Tp > Class Template Reference . . . 974
6.145.1 Detailed Description . . . 975
6.146 __gnu_cxx::_Relative_pointer_impl< const _Tp > Class Template Reference . . . 976
6.146.1 Detailed Description . . . 976
6.147 __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp > Class Template Reference . . . 976
6.147.1 Detailed Description . . . 977
6.147.2 Constructor & Destructor Documentation . . . 977
6.147.3 Member Function Documentation . . . 977
6.148 __gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware > Class Template Reference . . . 978
6.148.1 Detailed Description . . . 980
6.149 __gnu_debug::_Safe_forward_list< _SafeSequence > Class Template Reference . . . 980
6.149.1 Detailed Description . . . 980

6.149.2 Member Function Documentation . . . 981
6.149.3 Member Data Documentation . . . 981
6.150 __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category > Class Template Reference . . . 982
6.150.1 Detailed Description . . . 984
6.150.2 Constructor & Destructor Documentation . . . 984
6.150.3 Member Function Documentation . . . 985
6.150.4 Member Data Documentation . . . 991
6.151 __gnu_debug::_Safe_iterator_base Class Reference . . . 991
6.151.1 Detailed Description . . . 993
6.151.2 Constructor & Destructor Documentation . . . 993
6.151.3 Member Function Documentation . . . 994
6.151.4 Member Data Documentation . . . 995
6.152 __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer > Class Template Reference . . . 996
6.152.1 Detailed Description . . . 999
6.152.2 Constructor & Destructor Documentation . . . 999
6.152.3 Member Function Documentation . . . 1000
6.152.4 Member Data Documentation . . . 1005
6.153 __gnu_debug::_Safe_local_iterator_base Class Reference . . . 1005
6.153.1 Detailed Description . . . 1007
6.153.2 Constructor & Destructor Documentation . . . 1007
6.153.3 Member Function Documentation . . . 1008
6.153.4 Member Data Documentation . . . 1010
6.154 __gnu_debug::_Safe_node_sequence< _Sequence > Class Template Reference . . . 1010
6.154.1 Detailed Description . . . 1012
6.154.2 Member Function Documentation . . . 1012
6.154.3 Member Data Documentation . . . 1013
6.155 __gnu_debug::_Safe_sequence< _Sequence > Class Template Reference . . . 1013
6.155.1 Detailed Description . . . 1014
6.155.2 Member Function Documentation . . . 1015
6.155.3 Member Data Documentation . . . 1016
6.156 __gnu_debug::_Safe_sequence_base Class Reference . . . 1016
6.156.1 Detailed Description . . . 1018
6.156.2 Constructor & Destructor Documentation . . . 1018
6.156.3 Member Function Documentation . . . 1018
6.156.4 Member Data Documentation . . . 1019
6.157 __gnu_debug::_Safe_unordered_container< _Container > Class Template Reference . . . 1019
6.157.1 Detailed Description . . . 1020
6.157.2 Member Function Documentation . . . 1021
6.157.3 Member Data Documentation . . . 1022

6.158 __gnu_debug::_Safe_unordered_container_base Class Reference 1022
6.158.1 Detailed Description 1024
6.158.2 Constructor & Destructor Documentation 1024
6.158.3 Member Function Documentation 1024
6.158.4 Member Data Documentation 1025
6.159 __gnu_debug::_Safe_vector< _SafeSequence, _BaseSequence > Class Template Reference 1026
6.159.1 Detailed Description 1027
6.160 __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering > Struct Template Reference 1027
6.160.1 Detailed Description 1027
6.161 __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering > Struct Template Reference 1027
6.161.1 Detailed Description 1027
6.162 std::__detail::_Scanner< _CharT > Class Template Reference 1028
6.162.1 Detailed Description 1029
6.162.2 Member Enumeration Documentation 1029
6.163 __gnu_debug::_Sequence_traits< _Sequence > Struct Template Reference 1029
6.163.1 Detailed Description 1029
6.164 __gnu_parallel::_Settings Struct Reference 1029
6.164.1 Detailed Description 1031
6.164.2 Member Function Documentation 1031
6.164.3 Member Data Documentation 1031
6.165 std::_Sp_ebo_helper< _Tp, false > Struct Template Reference 1036
6.165.1 Detailed Description 1036
6.166 std::_Sp_ebo_helper< _Tp, true > Struct Template Reference 1036
6.166.1 Detailed Description 1037
6.167 __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference 1037
6.167.1 Detailed Description 1037
6.168 __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference 1037
6.168.1 Detailed Description 1037
6.169 __gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference 1037
6.169.1 Detailed Description 1038
6.170 std::__detail::_StateSeq< _TraitsT > Class Template Reference 1038
6.170.1 Detailed Description 1038
6.171 __gnu_cxx::_Std_pointer_impl< _Tp > Class Template Reference 1038
6.171.1 Detailed Description 1039
6.172 __gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference 1039
6.172.1 Detailed Description 1040
6.172.2 Constructor & Destructor Documentation 1040

6.172.3 Member Function Documentation . . . 1040
6.173 std::_Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference . . . 1040
6.173.1 Detailed Description . . . 1041
6.173.2 Constructor & Destructor Documentation . . . 1041
6.173.3 Member Function Documentation . . . 1042
6.174 __gnu_cxx::_Unqualified_type< _Tp > Struct Template Reference . . . 1042
6.174.1 Detailed Description . . . 1042
6.175 std::_Vector_base< _Tp, _Alloc > Struct Template Reference . . . 1043
6.175.1 Detailed Description . . . 1044
6.176 std::add_const< _Tp > Struct Template Reference . . . 1044
6.176.1 Detailed Description . . . 1045
6.177 std::add_cv< _Tp > Struct Template Reference . . . 1045
6.177.1 Detailed Description . . . 1045
6.178 std::add_lvalue_reference< _Tp > Struct Template Reference . . . 1045
6.178.1 Detailed Description . . . 1045
6.179 std::add_pointer< _Tp > Struct Template Reference . . . 1045
6.179.1 Detailed Description . . . 1046
6.180 std::add_rvalue_reference< _Tp > Struct Template Reference . . . 1046
6.180.1 Detailed Description . . . 1046
6.181 std::add_volatile< _Tp > Struct Template Reference . . . 1046
6.181.1 Detailed Description . . . 1046
6.182 std::adopt_lock_t Struct Reference . . . 1046
6.182.1 Detailed Description . . . 1046
6.183 std::aligned_storage< _Len, _Align > Struct Template Reference . . . 1047
6.183.1 Detailed Description . . . 1047
6.184 std::aligned_union< _Len, _Types > Struct Template Reference . . . 1047
6.184.1 Detailed Description . . . 1047
6.185 std::alignment_of< _Tp > Struct Template Reference . . . 1048
6.185.1 Detailed Description . . . 1048
6.186 __gnu_cxx::allocator< _Tp > Class Template Reference . . . 1048
6.186.1 Detailed Description . . . 1050
6.187 std::allocator< _Tp > Class Template Reference . . . 1050
6.187.1 Detailed Description . . . 1051
6.188 std::allocator< void > Class Reference . . . 1051
6.188.1 Detailed Description . . . 1052
6.188.2 Friends And Related Symbol Documentation . . . 1052
6.189 std::allocator_traits< _Alloc > Struct Template Reference . . . 1053
6.189.1 Detailed Description . . . 1055
6.189.2 Member Typedef Documentation . . . 1055

6.189.3 Member Function Documentation 1057
6.190 std::allocator_traits< allocator< _Tp > > Struct Template Reference 1060
6.190.1 Detailed Description 1061
6.190.2 Member Typedef Documentation 1061
6.190.3 Member Function Documentation 1062
6.191 std::allocator_traits< allocator< void > > Struct Reference 1065
6.191.1 Detailed Description 1066
6.191.2 Member Typedef Documentation 1066
6.191.3 Member Function Documentation 1067
6.192 std::allocator_traits< pmr::polymorphic_allocator< _Tp > > Struct Template Reference 1069
6.192.1 Detailed Description 1070
6.192.2 Member Typedef Documentation 1070
6.192.3 Member Function Documentation 1071
6.193 __gnu_cxx::limit_condition::always_adjustor Struct Reference 1074
6.193.1 Detailed Description 1074
6.194 __gnu_cxx::random_condition::always_adjustor Struct Reference 1074
6.194.1 Detailed Description 1074
6.195 __gnu_cxx::annotate_base Struct Reference 1074
6.195.1 Detailed Description 1075
6.196 std::experimental::fundamentals_v1::any Class Reference 1075
6.196.1 Detailed Description 1076
6.196.2 Constructor & Destructor Documentation 1076
6.196.3 Member Function Documentation 1077
6.197 std::array< _Tp, _Nm > Struct Template Reference 1078
6.197.1 Detailed Description 1079
6.198 __gnu_pbds::associative_tag Struct Reference 1080
6.198.1 Detailed Description 1080
6.199 std::atomic< _Tp > Struct Template Reference 1080
6.199.1 Detailed Description 1081
6.200 std::atomic< _Tp * > Struct Template Reference 1081
6.200.1 Detailed Description 1082
6.201 std::atomic< bool > Struct Reference 1082
6.201.1 Detailed Description 1083
6.202 std::atomic< char > Struct Reference 1083
6.202.1 Detailed Description 1084
6.203 std::atomic< char16_t > Struct Reference 1084
6.203.1 Detailed Description 1084
6.204 std::atomic< char32_t > Struct Reference 1085
6.204.1 Detailed Description 1085

6.205 std::atomic< int > Struct Reference . . . 1085
6.205.1 Detailed Description . . . 1086
6.206 std::atomic< long > Struct Reference . . . 1086
6.206.1 Detailed Description . . . 1086
6.207 std::atomic< long long > Struct Reference . . . 1086
6.207.1 Detailed Description . . . 1087
6.208 std::atomic< short > Struct Reference . . . 1087
6.208.1 Detailed Description . . . 1087
6.209 std::atomic< signed char > Struct Reference . . . 1088
6.209.1 Detailed Description . . . 1088
6.210 std::atomic< unsigned char > Struct Reference . . . 1088
6.210.1 Detailed Description . . . 1089
6.211 std::atomic< unsigned int > Struct Reference . . . 1089
6.211.1 Detailed Description . . . 1089
6.212 std::atomic< unsigned long > Struct Reference . . . 1089
6.212.1 Detailed Description . . . 1090
6.213 std::atomic< unsigned long long > Struct Reference . . . 1090
6.213.1 Detailed Description . . . 1091
6.214 std::atomic< unsigned short > Struct Reference . . . 1091
6.214.1 Detailed Description . . . 1091
6.215 std::atomic< wchar_t > Struct Reference . . . 1091
6.215.1 Detailed Description . . . 1092
6.216 std::atomic_flag Struct Reference . . . 1092
6.216.1 Detailed Description . . . 1092
6.217 std::auto_ptr< _Tp > Class Template Reference . . . 1092
6.217.1 Detailed Description . . . 1093
6.217.2 Member Typedef Documentation . . . 1093
6.217.3 Constructor & Destructor Documentation . . . 1094
6.217.4 Member Function Documentation . . . 1095
6.218 std::auto_ptr_ref< _Tp1 > Struct Template Reference . . . 1097
6.218.1 Detailed Description . . . 1097
6.219 std::back_insert_iterator< _Container > Class Template Reference . . . 1097
6.219.1 Detailed Description . . . 1098
6.219.2 Member Typedef Documentation . . . 1099
6.219.3 Constructor & Destructor Documentation . . . 1099
6.219.4 Member Function Documentation . . . 1099
6.220 std::bad_alloc Class Reference . . . 1100
6.220.1 Detailed Description . . . 1101
6.220.2 Member Function Documentation . . . 1101

6.221 std::experimental::fundamentals_v1::bad_any_cast Class Reference 1101
6.221.1 Detailed Description 1101
6.221.2 Member Function Documentation 1102
6.222 std::bad_cast Class Reference 1102
6.222.1 Detailed Description 1102
6.222.2 Member Function Documentation 1102
6.223 std::bad_exception Class Reference 1103
6.223.1 Detailed Description 1103
6.223.2 Member Function Documentation 1103
6.224 std::bad_function_call Class Reference 1103
6.224.1 Detailed Description 1104
6.224.2 Member Function Documentation 1104
6.225 std::experimental::fundamentals_v1::bad_optional_access Class Reference 1104
6.225.1 Detailed Description 1105
6.225.2 Member Function Documentation 1105
6.226 std::bad_typeid Class Reference 1105
6.226.1 Detailed Description 1106
6.226.2 Member Function Documentation 1106
6.227 std::bad_weak_ptr Class Reference 1106
6.227.1 Detailed Description 1107
6.227.2 Member Function Documentation 1107
6.228 __gnu_parallel::balanced_quicksort_tag Struct Reference 1107
6.228.1 Detailed Description 1108
6.228.2 Member Function Documentation 1108
6.229 __gnu_parallel::balanced_tag Struct Reference 1109
6.229.1 Detailed Description 1109
6.229.2 Member Function Documentation 1109
6.230 std::tr2::bases< _Tp > Struct Template Reference 1110
6.230.1 Detailed Description 1110
6.231 __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc > Class Template Reference 1110
6.231.1 Detailed Description 1111
6.232 __gnu_pbds::basic_branch_tag Struct Reference 1111
6.232.1 Detailed Description 1111
6.233 std::basic_filebuf< _CharT, _Traits > Class Template Reference 1111
6.233.1 Detailed Description 1115
6.233.2 Constructor & Destructor Documentation 1115
6.233.3 Member Function Documentation 1115
6.233.4 Member Data Documentation 1131

6.234 std::basic_fstream< _CharT, _Traits > Class Template Reference 1133
6.234.1 Detailed Description 1140
6.234.2 Member Typedef Documentation 1140
6.234.3 Member Enumeration Documentation 1142
6.234.4 Constructor & Destructor Documentation 1142
6.234.5 Member Function Documentation 1143
6.234.6 Member Data Documentation 1178
6.235 __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc > Class Template Reference 1183
6.235.1 Detailed Description 1184
6.236 __gnu_pbds::basic_hash_tag Struct Reference 1185
6.236.1 Detailed Description 1185
6.237 std::basic_ifstream< _CharT, _Traits > Class Template Reference 1186
6.237.1 Detailed Description 1191
6.237.2 Member Typedef Documentation 1191
6.237.3 Member Enumeration Documentation 1193
6.237.4 Constructor & Destructor Documentation 1193
6.237.5 Member Function Documentation 1194
6.237.6 Member Data Documentation 1222
6.238 __gnu_pbds::basic_invalidation_guarantee Struct Reference 1226
6.238.1 Detailed Description 1227
6.239 std::basic_ios< _CharT, _Traits > Class Template Reference 1227
6.239.1 Detailed Description 1231
6.239.2 Member Typedef Documentation 1231
6.239.3 Member Enumeration Documentation 1234
6.239.4 Constructor & Destructor Documentation 1234
6.239.5 Member Function Documentation 1235
6.239.6 Member Data Documentation 1246
6.240 std::basic_iostream< _CharT, _Traits > Class Template Reference 1251
6.240.1 Detailed Description 1257
6.240.2 Member Typedef Documentation 1257
6.240.3 Member Enumeration Documentation 1259
6.240.4 Constructor & Destructor Documentation 1259
6.240.5 Member Function Documentation 1259
6.240.6 Member Data Documentation 1293
6.241 std::basic_istream< _CharT, _Traits > Class Template Reference 1298
6.241.1 Detailed Description 1303
6.241.2 Member Typedef Documentation 1303
6.241.3 Member Enumeration Documentation 1305

6.241.4 Constructor & Destructor Documentation . . . 1306
6.241.5 Member Function Documentation . . . 1306
6.241.6 Member Data Documentation . . . 1332
6.242 std::basic_istringstream< _CharT, _Traits, _Alloc > Class Template Reference . . . 1337
6.242.1 Detailed Description . . . 1343
6.242.2 Member Typedef Documentation . . . 1343
6.242.3 Member Enumeration Documentation . . . 1345
6.242.4 Constructor & Destructor Documentation . . . 1346
6.242.5 Member Function Documentation . . . 1346
6.242.6 Member Data Documentation . . . 1373
6.243 std::basic_ofstream< _CharT, _Traits > Class Template Reference . . . 1378
6.243.1 Detailed Description . . . 1383
6.243.2 Member Typedef Documentation . . . 1383
6.243.3 Member Enumeration Documentation . . . 1385
6.243.4 Constructor & Destructor Documentation . . . 1385
6.243.5 Member Function Documentation . . . 1386
6.243.6 Member Data Documentation . . . 1407
6.244 std::basic_ostream< _CharT, _Traits > Class Template Reference . . . 1411
6.244.1 Detailed Description . . . 1416
6.244.2 Member Typedef Documentation . . . 1416
6.244.3 Member Enumeration Documentation . . . 1418
6.244.4 Constructor & Destructor Documentation . . . 1419
6.244.5 Member Function Documentation . . . 1419
6.244.6 Member Data Documentation . . . 1438
6.245 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference . . . 1443
6.245.1 Detailed Description . . . 1448
6.245.2 Member Typedef Documentation . . . 1449
6.245.3 Member Enumeration Documentation . . . 1451
6.245.4 Constructor & Destructor Documentation . . . 1451
6.245.5 Member Function Documentation . . . 1452
6.245.6 Member Data Documentation . . . 1471
6.246 std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference . . . 1476
6.246.1 Detailed Description . . . 1478
6.246.2 Constructor & Destructor Documentation . . . 1478
6.246.3 Member Function Documentation . . . 1481
6.247 std::basic_streambuf< _CharT, _Traits > Class Template Reference . . . 1486
6.247.1 Detailed Description . . . 1488
6.247.2 Member Typedef Documentation . . . 1489
6.247.3 Constructor & Destructor Documentation . . . 1490

6.247.4 Member Function Documentation . . . 1490
6.247.5 Member Data Documentation . . . 1503
6.248 __gnu_debug::basic_string< _CharT, _Traits, _Allocator > Class Template Reference . . . 1504
6.248.1 Detailed Description . . . 1511
6.248.2 Member Function Documentation . . . 1511
6.248.3 Member Data Documentation . . . 1581
6.249 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference . . . 1581
6.249.1 Detailed Description . . . 1590
6.249.2 Constructor & Destructor Documentation . . . 1591
6.249.3 Member Function Documentation . . . 1598
6.249.4 Member Data Documentation . . . 1686
6.250 std::basic_string_view< _CharT, _Traits > Class Template Reference . . . 1687
6.250.1 Detailed Description . . . 1689
6.251 std::experimental::filesystem::v1::basic_string_view< _CharT, _Traits > Class Template Reference . . . 1689
6.251.1 Detailed Description . . . 1691
6.252 std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits > Class Template Reference 1691
6.252.1 Detailed Description . . . 1693
6.253 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference . . . 1693
6.253.1 Detailed Description . . . 1696
6.253.2 Constructor & Destructor Documentation . . . 1696
6.253.3 Member Function Documentation . . . 1697
6.253.4 Member Data Documentation . . . 1711
6.254 std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template Reference . . . 1712
6.254.1 Detailed Description . . . 1718
6.254.2 Member Typedef Documentation . . . 1719
6.254.3 Member Enumeration Documentation . . . 1720
6.254.4 Constructor & Destructor Documentation . . . 1721
6.254.5 Member Function Documentation . . . 1721
6.254.6 Member Data Documentation . . . 1756
6.255 std::bernoulli_distribution Class Reference . . . 1760
6.255.1 Detailed Description . . . 1761
6.255.2 Member Typedef Documentation . . . 1761
6.255.3 Constructor & Destructor Documentation . . . 1761
6.255.4 Member Function Documentation . . . 1762
6.255.5 Friends And Related Symbol Documentation . . . 1763
6.256 std::bidirectional_iterator_tag Struct Reference . . . 1763
6.256.1 Detailed Description . . . 1764
6.257 __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference . . . 1764

6.257.1 Detailed Description . . . 1765
6.258 __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference . . . 1766
6.258.1 Detailed Description . . . 1766
6.258.2 Member Typedef Documentation . . . 1767
6.258.3 Member Function Documentation . . . 1767
6.259 __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference . . . 1768
6.259.1 Detailed Description . . . 1770
6.260 __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference . . . 1770
6.260.1 Detailed Description . . . 1771
6.260.2 Member Typedef Documentation . . . 1771
6.260.3 Member Function Documentation . . . 1772
6.261 __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference . . . 1773
6.261.1 Detailed Description . . . 1773
6.261.2 Member Typedef Documentation . . . 1774
6.262 __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference . . . 1774
6.262.1 Detailed Description . . . 1774
6.262.2 Member Typedef Documentation . . . 1774
6.263 __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 > Class Template Reference . . . 1775
6.263.1 Detailed Description . . . 1775
6.263.2 Member Typedef Documentation . . . 1775
6.264 std::binary_function< _Arg1, _Arg2, _Result > Struct Template Reference . . . 1776
6.264.1 Detailed Description . . . 1778
6.264.2 Member Typedef Documentation . . . 1778
6.265 __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference . . . 1778
6.265.1 Detailed Description . . . 1780
6.266 __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference . . . 1780
6.266.1 Detailed Description . . . 1781
6.266.2 Member Typedef Documentation . . . 1781
6.266.3 Constructor & Destructor Documentation . . . 1782
6.266.4 Member Function Documentation . . . 1782
6.267 __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference . . . 1783
6.267.1 Detailed Description . . . 1785
6.267.2 Member Typedef Documentation . . . 1785
6.267.3 Constructor & Destructor Documentation . . . 1786

6.267.4 Member Function Documentation . . . 1786
6.268 __gnu_pbds::binary_heap_tag Struct Reference . . . 1787
6.268.1 Detailed Description . . . 1787
6.269 std::binary_negate< _Predicate > Class Template Reference . . . 1787
6.269.1 Detailed Description . . . 1788
6.269.2 Member Typedef Documentation . . . 1788
6.270 std::binder1st< _Operation > Class Template Reference . . . 1788
6.270.1 Detailed Description . . . 1789
6.270.2 Member Typedef Documentation . . . 1789
6.271 std::binder2nd< _Operation > Class Template Reference . . . 1790
6.271.1 Detailed Description . . . 1790
6.271.2 Member Typedef Documentation . . . 1791
6.272 std::binomial_distribution< _IntType > Class Template Reference . . . 1791
6.272.1 Detailed Description . . . 1792
6.272.2 Member Typedef Documentation . . . 1792
6.272.3 Member Function Documentation . . . 1792
6.272.4 Friends And Related Symbol Documentation . . . 1793
6.273 __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference . . . 1794
6.273.1 Detailed Description . . . 1796
6.274 __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > Class Template Reference 1796
6.274.1 Detailed Description . . . 1798
6.275 __gnu_pbds::binomial_heap_tag Struct Reference . . . 1799
6.275.1 Detailed Description . . . 1799
6.276 __gnu_cxx::bitmap_allocator< _Tp > Class Template Reference . . . 1799
6.276.1 Detailed Description . . . 1800
6.276.2 Member Function Documentation . . . 1800
6.277 std::__debug::bitset< _Nb > Class Template Reference . . . 1801
6.277.1 Detailed Description . . . 1802
6.278 std::bitset< _Nb > Class Template Reference . . . 1802
6.278.1 Detailed Description . . . 1804
6.278.2 Constructor & Destructor Documentation . . . 1805
6.278.3 Member Function Documentation . . . 1807
6.279 std::tr2::bool_set Class Reference . . . 1813
6.279.1 Detailed Description . . . 1813
6.279.2 Constructor & Destructor Documentation . . . 1814
6.279.3 Member Function Documentation . . . 1814
6.280 __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc > Struct Template Reference . . . 1814
6.280.1 Detailed Description . . . 1815
6.281 __gnu_pbds::detail::branch_policy< Node_CItr, Node_CItr, _Alloc > Struct Template Reference . . . 1815

6.281.1 Detailed Description . . . 1816
6.282 std::cauchy_distribution< _RealType > Class Template Reference . . . 1816
6.282.1 Detailed Description . . . 1817
6.282.2 Member Typedef Documentation . . . 1817
6.282.3 Member Function Documentation . . . 1817
6.282.4 Friends And Related Symbol Documentation . . . 1818
6.283 __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference . . . 1818
6.283.1 Detailed Description . . . 1819
6.283.2 Member Enumeration Documentation . . . 1819
6.283.3 Constructor & Destructor Documentation . . . 1820
6.283.4 Member Function Documentation . . . 1820
6.284 __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc > Class Template Reference . . . 1822
6.284.1 Detailed Description . . . 1824
6.284.2 Constructor & Destructor Documentation . . . 1824
6.285 __gnu_pbds::cc_hash_tag Struct Reference . . . 1827
6.285.1 Detailed Description . . . 1828
6.286 __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash↩ _Fn, Resize_Policy > Class Template Reference . . . 1828
6.286.1 Detailed Description . . . 1831
6.286.2 Member Enumeration Documentation . . . 1831
6.286.3 Member Function Documentation . . . 1831
6.287 __gnu_cxx::char_traits< _CharT > Struct Template Reference . . . 1833
6.287.1 Detailed Description . . . 1834
6.288 std::char_traits< _CharT > Struct Template Reference . . . 1834
6.288.1 Detailed Description . . . 1835
6.289 std::char_traits< __gnu_cxx::character< _Value, _Int, _St > > Struct Template Reference . . . 1835
6.289.1 Detailed Description . . . 1837
6.290 std::char_traits< char > Struct Reference . . . 1837
6.290.1 Detailed Description . . . 1839
6.291 std::char_traits< wchar_t > Struct Reference . . . 1839
6.291.1 Detailed Description . . . 1841
6.292 __gnu_cxx::character< _Value, _Int, _St > Struct Template Reference . . . 1841
6.292.1 Detailed Description . . . 1841
6.293 std::chi_squared_distribution< _RealType > Class Template Reference . . . 1841
6.293.1 Detailed Description . . . 1842
6.293.2 Member Typedef Documentation . . . 1843
6.293.3 Member Function Documentation . . . 1843
6.293.4 Friends And Related Symbol Documentation . . . 1844

6.294 std::codecvt< _InternT, _ExternT, _StateT > Class Template Reference 1845
6.294.1 Detailed Description 1846
6.294.2 Member Function Documentation 1846
6.295 std::codecvt< _InternT, _ExternT, encoding_state > Class Template Reference 1850
6.295.1 Detailed Description 1852
6.295.2 Member Function Documentation 1852
6.296 std::codecvt< char, char, mbstate_t > Class Reference 1858
6.296.1 Detailed Description 1859
6.296.2 Member Function Documentation 1859
6.297 std::codecvt< char16_t, char, mbstate_t > Class Reference 1865
6.297.1 Detailed Description 1867
6.297.2 Member Function Documentation 1867
6.298 std::codecvt< char32_t, char, mbstate_t > Class Reference 1872
6.298.1 Detailed Description 1874
6.298.2 Member Function Documentation 1874
6.299 std::codecvt< wchar_t, char, mbstate_t > Class Reference 1880
6.299.1 Detailed Description 1882
6.299.2 Member Function Documentation 1882
6.300 std::codecvt_base Class Reference 1887
6.300.1 Detailed Description 1888
6.301 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference 1888
6.301.1 Detailed Description 1890
6.301.2 Member Function Documentation 1890
6.302 std::__cxx11::collate< _CharT > Class Template Reference 1894
6.302.1 Detailed Description 1895
6.302.2 Member Typedef Documentation 1895
6.302.3 Constructor & Destructor Documentation 1895
6.302.4 Member Function Documentation 1896
6.302.5 Member Data Documentation 1899
6.303 std::__cxx11::collate_byname< _CharT > Class Template Reference 1899
6.303.1 Detailed Description 1900
6.303.2 Member Typedef Documentation 1900
6.304 std::common_iterator< _It, _Sent > Class Template Reference 1900
6.304.1 Detailed Description 1901
6.305 std::common_type< _Tp > Struct Template Reference 1901
6.305.1 Detailed Description 1901
6.306 std::common_type< chrono::duration< _Rep, _Period > > Struct Template Reference 1901
6.306.1 Detailed Description 1901

6.307 std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > > Struct Template Reference . . . 1902
6.307.1 Detailed Description . . . 1902
6.308 std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > > Struct Template Reference . . . 1902
6.308.1 Detailed Description . . . 1902
6.309 std::common_type< chrono::time_point< _Clock, _Duration > > Struct Template Reference . . . 1902
6.309.1 Detailed Description . . . 1902
6.310 std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > > Struct Template Reference . . . 1902
6.310.1 Detailed Description . . . 1903
6.311 std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > > Struct Template Reference . . . 1903
6.311.1 Detailed Description . . . 1903
6.312 std::compare_three_way_result< _Tp, _Up > Struct Template Reference . . . 1903
6.312.1 Detailed Description . . . 1903
6.313 std::complex< _Tp > Class Template Reference . . . 1903
6.313.1 Detailed Description . . . 1904
6.313.2 Member Typedef Documentation . . . 1904
6.313.3 Constructor & Destructor Documentation . . . 1904
6.313.4 Member Function Documentation . . . 1905
6.314 std::complex< double > Class Reference . . . 1906
6.314.1 Detailed Description . . . 1907
6.315 std::complex< float > Class Reference . . . 1907
6.315.1 Detailed Description . . . 1908
6.316 std::complex< long double > Class Reference . . . 1908
6.316.1 Detailed Description . . . 1909
6.317 __gnu_pbds::detail::cond_dealtor< Entry, _Alloc > Class Template Reference . . . 1909
6.317.1 Detailed Description . . . 1909
6.318 __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type > Class Template Reference . . . 1910
6.318.1 Detailed Description . . . 1910
6.319 __gnu_cxx::condition_base Struct Reference . . . 1910
6.319.1 Detailed Description . . . 1911
6.320 std::condition_variable Class Reference . . . 1911
6.320.1 Detailed Description . . . 1911
6.321 std::condition_variable_any Class Reference . . . 1911
6.321.1 Detailed Description . . . 1912
6.322 std::conditional< _Cond, _Iftrue, _Iffalse > Struct Template Reference . . . 1912
6.322.1 Detailed Description . . . 1912

6.323 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator Struct Reference . . 1912
6.323.1 Detailed Description . . . 1913
6.324 std::chrono::tzdb_list::const_iterator Class Reference . . . 1914
6.324.1 Detailed Description . . . 1914
6.325 std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference . . . 1914
6.325.1 Detailed Description . . . 1915
6.325.2 Member Typedef Documentation . . . 1915
6.326 std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference . . . 1916
6.326.1 Detailed Description . . . 1916
6.326.2 Member Typedef Documentation . . . 1916
6.327 std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference . . . 1917
6.327.1 Detailed Description . . . 1917
6.327.2 Member Typedef Documentation . . . 1917
6.328 std::const_mem_fun_t< _Ret, _Tp > Class Template Reference . . . 1918
6.328.1 Detailed Description . . . 1918
6.328.2 Member Typedef Documentation . . . 1918
6.329 __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 > Struct Template Reference . . . 1919
6.329.1 Detailed Description . . . 1919
6.330 __gnu_parallel::constant_size_blocks_tag Struct Reference . . . 1919
6.330.1 Detailed Description . . . 1920
6.331 __gnu_cxx::constant_unary_fun< _Result, _Argument > Struct Template Reference . . . 1920
6.331.1 Detailed Description . . . 1920
6.332 __gnu_cxx::constant_void_fun< _Result > Struct Template Reference . . . 1921
6.332.1 Detailed Description . . . 1921
6.333 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_Tl > Struct Template Reference . . . 1921
6.333.1 Detailed Description . . . 1921
6.334 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type > Struct Template Reference . . . 1921
6.334.1 Detailed Description . . . 1921
6.334.2 Member Typedef Documentation . . . 1922
6.335 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type > Struct Template Reference . . . 1922
6.335.1 Detailed Description . . . 1922
6.335.2 Member Typedef Documentation . . . 1922
6.336 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type > Struct Template Reference . . . 1922
6.336.1 Detailed Description . . . 1922
6.336.2 Member Typedef Documentation . . . 1923

6.337 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type > Struct Template Reference . . . 1923
6.337.1 Detailed Description . . . 1923
6.337.2 Member Typedef Documentation . . . 1923
6.338 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type > Struct Template Reference . . . 1923
6.338.1 Detailed Description . . . 1923
6.338.2 Member Typedef Documentation . . . 1924
6.339 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl > Struct Template Reference . . . 1924
6.339.1 Detailed Description . . . 1924
6.339.2 Member Typedef Documentation . . . 1924
6.340 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl > Struct Template Reference . . . 1924
6.340.1 Detailed Description . . . 1924
6.340.2 Member Typedef Documentation . . . 1925
6.341 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl > Struct Template Reference . . . 1925
6.341.1 Detailed Description . . . 1925
6.341.2 Member Typedef Documentation . . . 1925
6.342 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl > Struct Template Reference . . . 1925
6.342.1 Detailed Description . . . 1925
6.342.2 Member Typedef Documentation . . . 1926
6.343 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl > Struct Template Reference . . . 1926
6.343.1 Detailed Description . . . 1926
6.344 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl > Struct Template Reference . . . 1926
6.344.1 Detailed Description . . . 1926
6.344.2 Member Typedef Documentation . . . 1926
6.345 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl > Struct Template Reference . . . 1927
6.345.1 Detailed Description . . . 1927
6.345.2 Member Typedef Documentation . . . 1927
6.346 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl > Struct Template Reference . . . 1927
6.346.1 Detailed Description . . . 1927
6.346.2 Member Typedef Documentation . . . 1927
6.347 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl > Struct Template Reference . . . 1928
6.347.1 Detailed Description . . . 1928

6.347.2 Member Typedef Documentation . . . 1928
6.348 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl > Struct Template Reference . . . 1928
6.348.1 Detailed Description . . . 1928
6.348.2 Member Typedef Documentation . . . 1928
6.349 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl > Struct Template Reference . . . 1929
6.349.1 Detailed Description . . . 1929
6.349.2 Member Typedef Documentation . . . 1929
6.350 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl > Struct Template Reference . . . 1929
6.350.1 Detailed Description . . . 1929
6.350.2 Member Typedef Documentation . . . 1929
6.351 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl > Struct Template Reference . . . 1930
6.351.1 Detailed Description . . . 1930
6.352 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl > Struct Template Reference . . . 1930
6.352.1 Detailed Description . . . 1930
6.352.2 Member Typedef Documentation . . . 1930
6.353 __gnu_pbds::container_error Struct Reference . . . 1930
6.353.1 Detailed Description . . . 1931
6.353.2 Member Function Documentation . . . 1931
6.354 __gnu_pbds::container_tag Struct Reference . . . 1931
6.354.1 Detailed Description . . . 1932
6.355 __gnu_pbds::container_traits< Cntnr > Struct Template Reference . . . 1932
6.355.1 Detailed Description . . . 1933
6.355.2 Member Enumeration Documentation . . . 1933
6.356 __gnu_pbds::container_traits_base< _Tag > Struct Template Reference . . . 1933
6.356.1 Detailed Description . . . 1933
6.357 __gnu_pbds::container_traits_base< binary_heap_tag > Struct Reference . . . 1933
6.357.1 Detailed Description . . . 1934
6.358 __gnu_pbds::container_traits_base< binomial_heap_tag > Struct Reference . . . 1934
6.358.1 Detailed Description . . . 1934
6.359 __gnu_pbds::container_traits_base< cc_hash_tag > Struct Reference . . . 1934
6.359.1 Detailed Description . . . 1934
6.360 __gnu_pbds::container_traits_base< gp_hash_tag > Struct Reference . . . 1934
6.360.1 Detailed Description . . . 1934
6.361 __gnu_pbds::container_traits_base< list_update_tag > Struct Reference . . . 1935
6.361.1 Detailed Description . . . 1935

6.362 __gnu_pbds::container_traits_base< ov_tree_tag > Struct Reference . . . 1935
6.362.1 Detailed Description . . . 1935
6.363 __gnu_pbds::container_traits_base< pairing_heap_tag > Struct Reference . . . 1935
6.363.1 Detailed Description . . . 1935
6.364 __gnu_pbds::container_traits_base< pat_trie_tag > Struct Reference . . . 1935
6.364.1 Detailed Description . . . 1936
6.365 __gnu_pbds::container_traits_base< rb_tree_tag > Struct Reference . . . 1936
6.365.1 Detailed Description . . . 1936
6.366 __gnu_pbds::container_traits_base< rc_binomial_heap_tag > Struct Reference . . . 1936
6.366.1 Detailed Description . . . 1936
6.367 __gnu_pbds::container_traits_base< splay_tree_tag > Struct Reference . . . 1936
6.367.1 Detailed Description . . . 1937
6.368 __gnu_pbds::container_traits_base< thin_heap_tag > Struct Reference . . . 1937
6.368.1 Detailed Description . . . 1937
6.369 std::contiguous_iterator_tag Struct Reference . . . 1937
6.369.1 Detailed Description . . . 1938
6.370 std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference . . . 1938
6.370.1 Detailed Description . . . 1939
6.370.2 Constructor & Destructor Documentation . . . 1939
6.370.3 Member Function Documentation . . . 1941
6.370.4 Friends And Related Symbol Documentation . . . 1942
6.371 std::counted_iterator< _It > Class Template Reference . . . 1942
6.371.1 Detailed Description . . . 1943
6.372 std::ctype< _CharT > Class Template Reference . . . 1944
6.372.1 Detailed Description . . . 1945
6.372.2 Member Function Documentation . . . 1946
6.372.3 Member Data Documentation . . . 1957
6.373 std::ctype< char > Class Reference . . . 1957
6.373.1 Detailed Description . . . 1960
6.373.2 Member Typedef Documentation . . . 1960
6.373.3 Constructor & Destructor Documentation . . . 1961
6.373.4 Member Function Documentation . . . 1961
6.373.5 Member Data Documentation . . . 1987
6.374 std::ctype< wchar_t > Class Reference . . . 1988
6.374.1 Detailed Description . . . 1990
6.374.2 Member Typedef Documentation . . . 1990
6.374.3 Constructor & Destructor Documentation . . . 1990
6.374.4 Member Function Documentation . . . 1991

6.374.5 Member Data Documentation . . . 2001
6.375 std::ctype_base Struct Reference . . . 2001
6.375.1 Detailed Description . . . 2002
6.376 std::ctype_byname< _CharT > Class Template Reference . . . 2002
6.376.1 Detailed Description . . . 2004
6.376.2 Member Function Documentation . . . 2005
6.376.3 Member Data Documentation . . . 2016
6.377 std::ctype_byname< char > Class Reference . . . 2016
6.377.1 Detailed Description . . . 2019
6.377.2 Member Function Documentation . . . 2019
6.377.3 Member Data Documentation . . . 2039
6.378 std::ranges::dangling Struct Reference . . . 2040
6.378.1 Detailed Description . . . 2040
6.379 __gnu_cxx::debug_allocator< _Alloc > Class Template Reference . . . 2040
6.379.1 Detailed Description . . . 2041
6.380 std::decimal::decimal128 Class Reference . . . 2041
6.380.1 Detailed Description . . . 2042
6.380.2 Constructor & Destructor Documentation . . . 2042
6.381 std::decimal::decimal32 Class Reference . . . 2042
6.381.1 Detailed Description . . . 2044
6.381.2 Constructor & Destructor Documentation . . . 2044
6.382 std::decimal::decimal64 Class Reference . . . 2044
6.382.1 Detailed Description . . . 2045
6.382.2 Constructor & Destructor Documentation . . . 2045
6.383 parallelism_v2::simd_abi::deduce< _Tp, _Np,... > Struct Template Reference . . . 2045
6.383.1 Detailed Description . . . 2046
6.384 __gnu_pbds::detail::default_comb_hash_fn Struct Reference . . . 2046
6.384.1 Detailed Description . . . 2046
6.384.2 Member Typedef Documentation . . . 2046
6.385 std::default_delete< _Tp > Struct Template Reference . . . 2046
6.385.1 Detailed Description . . . 2046
6.385.2 Constructor & Destructor Documentation . . . 2047
6.385.3 Member Function Documentation . . . 2047
6.386 std::default_delete< _Tp[]> Struct Template Reference . . . 2047
6.386.1 Detailed Description . . . 2047
6.386.2 Constructor & Destructor Documentation . . . 2048
6.386.3 Member Function Documentation . . . 2048
6.387 __gnu_pbds::detail::default_eq_fn< Key > Struct Template Reference . . . 2048
6.387.1 Detailed Description . . . 2048

6.387.2 Member Typedef Documentation . . . 2048
6.388 __gnu_pbds::detail::default_hash_fn< Key > Struct Template Reference . . . 2049
6.388.1 Detailed Description . . . 2049
6.388.2 Member Typedef Documentation . . . 2049
6.389 __gnu_parallel::default_parallel_tag Struct Reference . . . 2049
6.389.1 Detailed Description . . . 2050
6.389.2 Member Function Documentation . . . 2050
6.390 __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn > Struct Template Reference . . . 2050
6.390.1 Detailed Description . . . 2050
6.390.2 Member Typedef Documentation . . . 2050
6.391 __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn > Struct Template Reference . . . 2051
6.391.1 Detailed Description . . . 2051
6.391.2 Member Typedef Documentation . . . 2051
6.392 std::default_sentinel_t Struct Reference . . . 2051
6.392.1 Detailed Description . . . 2051
6.393 __gnu_pbds::detail::default_trie_access_traits< Key > Struct Template Reference . . . 2051
6.393.1 Detailed Description . . . 2051
6.394 __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > > Struct Template Reference . . . 2052
6.394.1 Detailed Description . . . 2052
6.394.2 Member Typedef Documentation . . . 2052
6.395 __gnu_pbds::detail::default_update_policy Struct Reference . . . 2052
6.395.1 Detailed Description . . . 2052
6.395.2 Member Typedef Documentation . . . 2052
6.396 std::defer_lock_t Struct Reference . . . 2052
6.396.1 Detailed Description . . . 2053
6.397 std::__debug::deque< _Tp, _Allocator > Class Template Reference . . . 2053
6.397.1 Detailed Description . . . 2055
6.398 std::deque< _Tp, _Alloc > Class Template Reference . . . 2055
6.398.1 Detailed Description . . . 2059
6.398.2 Constructor & Destructor Documentation . . . 2060
6.398.3 Member Function Documentation . . . 2063
6.399 std::destroying_delete_t Struct Reference . . . 2077
6.399.1 Detailed Description . . . 2077
6.400 std::tr2::direct_bases< _Tp > Struct Template Reference . . . 2078
6.400.1 Detailed Description . . . 2078
6.401 __gnu_pbds::direct_mask_range_hashing< Size_Type > Class Template Reference . . . 2078
6.401.1 Detailed Description . . . 2079
6.401.2 Member Function Documentation . . . 2079

6.402 __gnu_pbds::direct_mod_range_hashing< Size_Type > Class Template Reference 2079
6.402.1 Detailed Description 2080
6.402.2 Member Function Documentation 2080
6.403 std::filesystem::directory_entry Class Reference 2080
6.403.1 Detailed Description 2081
6.404 std::filesystem::directory_iterator Class Reference 2081
6.404.1 Detailed Description 2082
6.405 std::discard_block_engine< _RandomNumberEngine, __p, __r > Class Template Reference 2083
6.405.1 Detailed Description 2083
6.405.2 Member Typedef Documentation 2084
6.405.3 Constructor & Destructor Documentation 2084
6.405.4 Member Function Documentation 2085
6.405.5 Friends And Related Symbol Documentation 2086
6.406 std::discrete_distribution< _IntType > Class Template Reference 2087
6.406.1 Detailed Description 2088
6.406.2 Member Typedef Documentation 2089
6.406.3 Member Function Documentation 2089
6.406.4 Friends And Related Symbol Documentation 2090
6.407 std::divides< _Tp > Struct Template Reference 2091
6.407.1 Detailed Description 2091
6.407.2 Member Typedef Documentation 2092
6.408 std::divides< void > Struct Reference 2092
6.408.1 Detailed Description 2093
6.408.2 Member Typedef Documentation 2093
6.409 std::domain_error Class Reference 2093
6.409.1 Detailed Description 2094
6.409.2 Member Function Documentation 2094
6.410 __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc > Struct Template Reference 2094
6.410.1 Detailed Description 2094
6.411 std::chrono::duration< _Rep, _Period > Class Template Reference 2094
6.411.1 Detailed Description 2095
6.412 std::chrono::duration_values< _Rep > Struct Template Reference 2096
6.412.1 Detailed Description 2096
6.413 std::tr2::dynamic_bitset< _WordT, _Alloc > Class Template Reference 2096
6.413.1 Detailed Description 2098
6.413.2 Constructor & Destructor Documentation 2099
6.413.3 Member Function Documentation 2101
6.414 std::ranges::elements_of< _Range, _Alloc > Struct Template Reference 2110
6.414.1 Detailed Description 2110

6.415 std::ranges::empty_view< _Tp > Class Template Reference 2110
6.415.1 Detailed Description 2111
6.416 std::enable_if< bool, _Tp > Struct Template Reference 2111
6.416.1 Detailed Description 2111
6.417 std::enable_shared_from_this< _Tp > Class Template Reference 2111
6.417.1 Detailed Description 2112
6.418 __gnu_cxx::enc_filebuf< _CharT > Class Template Reference 2112
6.418.1 Detailed Description 2115
6.418.2 Member Function Documentation 2115
6.418.3 Member Data Documentation 2127
6.419 __gnu_cxx::encoding_char_traits< _CharT > Struct Template Reference 2130
6.419.1 Detailed Description 2131
6.420 __gnu_cxx::encoding_state Class Reference 2131
6.420.1 Detailed Description 2132
6.421 std::encoding_state Class Reference 2132
6.421.1 Detailed Description 2133
6.422 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw > Struct Template Reference . . 2133
6.422.1 Detailed Description 2133
6.423 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false > Struct Template Reference 2133
6.423.1 Detailed Description 2133
6.424 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true > Struct Template Reference 2133
6.424.1 Detailed Description 2133
6.424.2 Member Typedef Documentation 2134
6.425 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw > Struct Template Reference 2134
6.425.1 Detailed Description 2134
6.426 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false > Struct Template Reference 2134
6.426.1 Detailed Description 2134
6.427 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true > Struct Template Reference 2134
6.427.1 Detailed Description 2134
6.428 __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn > Struct Template Reference 2135
6.428.1 Detailed Description 2135
6.429 __gnu_parallel::equal_split_tag Struct Reference 2135
6.429.1 Detailed Description 2135
6.430 __gnu_cxx::equal_to< _Tp > Struct Template Reference 2136
6.430.1 Detailed Description 2136
6.430.2 Member Typedef Documentation 2136
6.431 std::equal_to< _Tp > Struct Template Reference 2137
6.431.1 Detailed Description 2137
6.431.2 Member Typedef Documentation 2137

6.432 std::ranges::equal_to Struct Reference . . . 2138
6.432.1 Detailed Description . . . 2138
6.433 std::equal_to< void > Struct Reference . . . 2138
6.433.1 Detailed Description . . . 2139
6.433.2 Member Typedef Documentation . . . 2139
6.434 std::error_category Class Reference . . . 2139
6.434.1 Detailed Description . . . 2140
6.434.2 Member Function Documentation . . . 2140
6.435 std::error_code Class Reference . . . 2141
6.435.1 Detailed Description . . . 2142
6.435.2 Constructor & Destructor Documentation . . . 2142
6.435.3 Member Function Documentation . . . 2142
6.436 std::error_condition Class Reference . . . 2143
6.436.1 Detailed Description . . . 2143
6.436.2 Constructor & Destructor Documentation . . . 2143
6.436.3 Member Function Documentation . . . 2144
6.437 __gnu_parallel::exact_tag Struct Reference . . . 2144
6.437.1 Detailed Description . . . 2145
6.437.2 Member Function Documentation . . . 2145
6.438 std::exception Class Reference . . . 2146
6.438.1 Detailed Description . . . 2146
6.438.2 Member Function Documentation . . . 2147
6.439 std::__unspecified__::exception_ptr Class Reference . . . 2147
6.439.1 Detailed Description . . . 2147
6.440 std::exception_ptr Class Reference . . . 2148
6.440.1 Detailed Description . . . 2148
6.441 std::exponential_distribution< _RealType > Class Template Reference . . . 2148
6.441.1 Detailed Description . . . 2149
6.441.2 Member Typedef Documentation . . . 2150
6.441.3 Constructor & Destructor Documentation . . . 2150
6.441.4 Member Function Documentation . . . 2150
6.441.5 Friends And Related Symbol Documentation . . . 2151
6.442 std::extent< typename, _Uint > Struct Template Reference . . . 2151
6.442.1 Detailed Description . . . 2152
6.443 std::extreme_value_distribution< _RealType > Class Template Reference . . . 2152
6.443.1 Detailed Description . . . 2153
6.443.2 Member Typedef Documentation . . . 2153
6.443.3 Member Function Documentation . . . 2153
6.443.4 Friends And Related Symbol Documentation . . . 2155

6.444 std::locale::facet Class Reference . . . 2155
6.444.1 Detailed Description . . . 2157
6.444.2 Constructor & Destructor Documentation . . . 2157
6.445 std::ios_base::failure Class Reference . . . 2158
6.445.1 Detailed Description . . . 2158
6.445.2 Member Function Documentation . . . 2158
6.446 std::filesystem::file_status Class Reference . . . 2159
6.446.1 Detailed Description . . . 2159
6.447 std::experimental::filesystem::v1::filesystem_error Class Reference . . . 2159
6.447.1 Detailed Description . . . 2160
6.447.2 Member Function Documentation . . . 2160
6.448 std::filesystem::filesystem_error Class Reference . . . 2161
6.448.1 Detailed Description . . . 2161
6.448.2 Member Function Documentation . . . 2162
6.449 __gnu_parallel::find_tag Struct Reference . . . 2162
6.449.1 Detailed Description . . . 2162
6.450 std::fisher_f_distribution< _RealType > Class Template Reference . . . 2162
6.450.1 Detailed Description . . . 2163
6.450.2 Member Typedef Documentation . . . 2164
6.450.3 Member Function Documentation . . . 2164
6.450.4 Friends And Related Symbol Documentation . . . 2165
6.451 __gnu_cxx::forced_error Struct Reference . . . 2166
6.451.1 Detailed Description . . . 2166
6.451.2 Member Function Documentation . . . 2166
6.452 std::forward_iterator_tag Struct Reference . . . 2166
6.452.1 Detailed Description . . . 2167
6.453 std::__debug::forward_list< _Tp, _Alloc > Class Template Reference . . . 2167
6.453.1 Detailed Description . . . 2170
6.454 std::forward_list< _Tp, _Alloc > Class Template Reference . . . 2170
6.454.1 Detailed Description . . . 2172
6.454.2 Constructor & Destructor Documentation . . . 2172
6.454.3 Member Function Documentation . . . 2175
6.455 std::fpos< _StateT > Class Template Reference . . . 2188
6.455.1 Detailed Description . . . 2188
6.455.2 Constructor & Destructor Documentation . . . 2189
6.455.3 Member Function Documentation . . . 2189
6.456 __gnu_cxx::free_list Class Reference . . . 2190
6.456.1 Detailed Description . . . 2190
6.456.2 Member Function Documentation . . . 2191

6.457 std::from_chars_result Struct Reference . . . 2191
6.457.1 Detailed Description . . . 2192
6.458 std::front_insert_iterator< _Container > Class Template Reference . . . 2192
6.458.1 Detailed Description . . . 2193
6.458.2 Member Typedef Documentation . . . 2193
6.458.3 Constructor & Destructor Documentation . . . 2193
6.458.4 Member Function Documentation . . . 2193
6.459 std::function< _Res(_ArgTypes...)> Class Template Reference . . . 2194
6.459.1 Detailed Description . . . 2195
6.459.2 Constructor & Destructor Documentation . . . 2195
6.459.3 Member Function Documentation . . . 2197
6.460 std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference . . . 2200
6.460.1 Detailed Description . . . 2201
6.460.2 Constructor & Destructor Documentation . . . 2201
6.460.3 Member Function Documentation . . . 2202
6.461 std::future< _Res > Class Template Reference . . . 2202
6.461.1 Detailed Description . . . 2204
6.461.2 Constructor & Destructor Documentation . . . 2204
6.461.3 Member Function Documentation . . . 2204
6.462 std::future< _Res & > Class Template Reference . . . 2204
6.462.1 Detailed Description . . . 2205
6.462.2 Constructor & Destructor Documentation . . . 2205
6.462.3 Member Function Documentation . . . 2205
6.463 std::future< void > Class Reference . . . 2206
6.463.1 Detailed Description . . . 2207
6.463.2 Constructor & Destructor Documentation . . . 2207
6.463.3 Member Function Documentation . . . 2207
6.464 std::future_error Class Reference . . . 2207
6.464.1 Detailed Description . . . 2208
6.464.2 Member Function Documentation . . . 2208
6.465 std::gamma_distribution< _RealType > Class Template Reference . . . 2208
6.465.1 Detailed Description . . . 2209
6.465.2 Member Typedef Documentation . . . 2210
6.465.3 Constructor & Destructor Documentation . . . 2210
6.465.4 Member Function Documentation . . . 2210
6.465.5 Friends And Related Symbol Documentation . . . 2211
6.466 std::geometric_distribution< _IntType > Class Template Reference . . . 2212
6.466.1 Detailed Description . . . 2213

6.466.2 Member Typedef Documentation . . . 2213
6.466.3 Member Function Documentation . . . 2214
6.466.4 Friends And Related Symbol Documentation . . . 2215
6.467 __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc > Class Template Reference . . . 2215
6.467.1 Detailed Description . . . 2216
6.467.2 Constructor & Destructor Documentation . . . 2216
6.468 __gnu_pbds::gp_hash_tag Struct Reference . . . 2220
6.468.1 Detailed Description . . . 2221
6.469 __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > Class Template Reference . . . 2221
6.469.1 Detailed Description . . . 2224
6.469.2 Member Enumeration Documentation . . . 2224
6.469.3 Member Function Documentation . . . 2224
6.470 std::chrono::gps_clock Class Reference . . . 2226
6.470.1 Detailed Description . . . 2226
6.471 std::greater< _Tp > Struct Template Reference . . . 2227
6.471.1 Detailed Description . . . 2227
6.471.2 Member Typedef Documentation . . . 2227
6.472 std::ranges::greater Struct Reference . . . 2228
6.472.1 Detailed Description . . . 2228
6.473 std::greater< void > Struct Reference . . . 2228
6.473.1 Detailed Description . . . 2229
6.473.2 Member Typedef Documentation . . . 2229
6.474 std::greater_equal< _Tp > Struct Template Reference . . . 2229
6.474.1 Detailed Description . . . 2230
6.474.2 Member Typedef Documentation . . . 2230
6.475 std::ranges::greater_equal Struct Reference . . . 2230
6.475.1 Detailed Description . . . 2230
6.476 std::greater_equal< void > Struct Reference . . . 2230
6.476.1 Detailed Description . . . 2231
6.476.2 Member Typedef Documentation . . . 2231
6.477 __gnu_cxx::random_condition::group_adjustor Struct Reference . . . 2232
6.477.1 Detailed Description . . . 2232
6.478 __gnu_parallel::growing_blocks_tag Struct Reference . . . 2232
6.478.1 Detailed Description . . . 2232
6.479 std::gslice Class Reference . . . 2232
6.479.1 Detailed Description . . . 2233
6.480 std::gslice_array< _Tp > Class Template Reference . . . 2233
6.480.1 Detailed Description . . . 2234

6.480.2 Member Function Documentation . . . 2234
6.481 std::has_virtual_destructor< _Tp > Struct Template Reference . . . 2236
6.481.1 Detailed Description . . . 2236
6.482 std::hash< _Tp > Struct Template Reference . . . 2236
6.482.1 Detailed Description . . . 2236
6.483 std::hash< __debug::bitset< _Nb > > Struct Template Reference . . . 2236
6.483.1 Detailed Description . . . 2236
6.484 std::hash< __debug::vector< bool, _Alloc > > Struct Template Reference . . . 2237
6.484.1 Detailed Description . . . 2237
6.485 std::hash< __gnu_cxx::__u16vstring > Struct Reference . . . 2237
6.485.1 Detailed Description . . . 2237
6.486 std::hash< __gnu_cxx::__u32vstring > Struct Reference . . . 2237
6.486.1 Detailed Description . . . 2237
6.487 std::hash< __gnu_cxx::__vstring > Struct Reference . . . 2237
6.487.1 Detailed Description . . . 2238
6.488 std::hash< __gnu_cxx::__wvstring > Struct Reference . . . 2238
6.488.1 Detailed Description . . . 2238
6.489 std::hash< __gnu_cxx::throw_value_limit > Struct Reference . . . 2238
6.489.1 Detailed Description . . . 2239
6.489.2 Member Typedef Documentation . . . 2239
6.490 std::hash< __gnu_cxx::throw_value_random > Struct Reference . . . 2239
6.490.1 Detailed Description . . . 2239
6.490.2 Member Typedef Documentation . . . 2240
6.491 std::hash< __gnu_debug::basic_string< _CharT > > Struct Template Reference . . . 2240
6.491.1 Detailed Description . . . 2240
6.492 std::hash< __shared_ptr< _Tp, _Lp > > Struct Template Reference . . . 2240
6.492.1 Detailed Description . . . 2241
6.493 std::hash< _Tp * > Struct Template Reference . . . 2241
6.493.1 Detailed Description . . . 2241
6.494 std::hash< basic_string< char, char_traits< char >, _Alloc > > Struct Template Reference . . . 2241
6.494.1 Detailed Description . . . 2241
6.495 std::hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > > Struct Template Reference . . . 2241
6.495.1 Detailed Description . . . 2242
6.496 std::hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > > Struct Template Reference . . . 2242
6.496.1 Detailed Description . . . 2242
6.497 std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > > Struct Template Reference . . . 2242
6.497.1 Detailed Description . . . 2242
6.498 std::hash< bool > Struct Reference . . . 2242
6.498.1 Detailed Description . . . 2243

6.499 std::hash< char > Struct Reference . . . 2243
6.499.1 Detailed Description . . . 2243
6.500 std::hash< char16_t > Struct Reference . . . 2243
6.500.1 Detailed Description . . . 2243
6.501 std::hash< char32_t > Struct Reference . . . 2243
6.501.1 Detailed Description . . . 2243
6.502 std::hash< double > Struct Reference . . . 2243
6.502.1 Detailed Description . . . 2244
6.503 std::hash< error_code > Struct Reference . . . 2244
6.503.1 Detailed Description . . . 2244
6.504 std::hash< error_condition > Struct Reference . . . 2244
6.504.1 Detailed Description . . . 2244
6.505 std::hash< experimental::optional< _Tp > > Struct Template Reference . . . 2244
6.505.1 Detailed Description . . . 2245
6.506 std::hash< experimental::shared_ptr< _Tp > > Struct Template Reference . . . 2245
6.506.1 Detailed Description . . . 2245
6.507 std::hash< float > Struct Reference . . . 2245
6.507.1 Detailed Description . . . 2245
6.508 std::hash< int > Struct Reference . . . 2245
6.508.1 Detailed Description . . . 2245
6.509 std::hash< long > Struct Reference . . . 2246
6.509.1 Detailed Description . . . 2246
6.510 std::hash< long double > Struct Reference . . . 2246
6.510.1 Detailed Description . . . 2246
6.511 std::hash< long long > Struct Reference . . . 2246
6.511.1 Detailed Description . . . 2246
6.512 std::hash< shared_ptr< _Tp > > Struct Template Reference . . . 2246
6.512.1 Detailed Description . . . 2247
6.513 std::hash< short > Struct Reference . . . 2247
6.513.1 Detailed Description . . . 2247
6.514 std::hash< signed char > Struct Reference . . . 2247
6.514.1 Detailed Description . . . 2247
6.515 std::hash< thread::id > Struct Reference . . . 2247
6.515.1 Detailed Description . . . 2247
6.516 std::hash< type_index > Struct Reference . . . 2247
6.516.1 Detailed Description . . . 2248
6.517 std::hash< unique_ptr< _Tp, _Dp > > Struct Template Reference . . . 2248
6.517.1 Detailed Description . . . 2248
6.518 std::hash< unsigned char > Struct Reference . . . 2248

6.518.1 Detailed Description . . . 2248
6.519 std::hash< unsigned int > Struct Reference . . . 2248
6.519.1 Detailed Description . . . 2248
6.520 std::hash< unsigned long > Struct Reference . . . 2249
6.520.1 Detailed Description . . . 2249
6.521 std::hash< unsigned long long > Struct Reference . . . 2249
6.521.1 Detailed Description . . . 2249
6.522 std::hash< unsigned short > Struct Reference . . . 2249
6.522.1 Detailed Description . . . 2249
6.523 std::hash< wchar_t > Struct Reference . . . 2249
6.523.1 Detailed Description . . . 2249
6.524 std::hash<::bitset< _Nb > > Struct Template Reference . . . 2250
6.524.1 Detailed Description . . . 2250
6.525 std::hash<::vector< bool, _Alloc > > Struct Template Reference . . . 2250
6.525.1 Detailed Description . . . 2250
6.526 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash > Struct Template Reference . . . 2251
6.526.1 Detailed Description . . . 2251
6.527 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false > Struct Template Reference . . . 2251
6.527.1 Detailed Description . . . 2252
6.528 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true > Struct Template Reference . . . 2252
6.528.1 Detailed Description . . . 2252
6.529 __gnu_pbds::hash_exponential_size_policy< Size_Type > Class Template Reference . . . 2252
6.529.1 Detailed Description . . . 2253
6.529.2 Constructor & Destructor Documentation . . . 2253
6.530 __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference . . . 2253
6.530.1 Detailed Description . . . 2254
6.530.2 Member Enumeration Documentation . . . 2255
6.530.3 Constructor & Destructor Documentation . . . 2255
6.530.4 Member Function Documentation . . . 2255
6.531 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size > Class Template Reference . . . 2256
6.531.1 Detailed Description . . . 2256
6.532 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true > Class Template Reference . . . 2256
6.532.1 Detailed Description . . . 2257
6.533 __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference . . . 2257
6.533.1 Detailed Description . . . 2258
6.534 __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc > Class Template Reference . . . 2259
6.534.1 Detailed Description . . . 2260

6.535 __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference . . . 2260
6.535.1 Detailed Description . . . 2261
6.536 __gnu_pbds::hash_prime_size_policy Class Reference . . . 2262
6.536.1 Detailed Description . . . 2262
6.536.2 Member Typedef Documentation . . . 2262
6.536.3 Constructor & Destructor Documentation . . . 2262
6.537 __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference . . . 2262
6.537.1 Detailed Description . . . 2264
6.538 __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > Class Template Reference . . . 2264
6.538.1 Detailed Description . . . 2265
6.538.2 Member Enumeration Documentation . . . 2265
6.538.3 Constructor & Destructor Documentation . . . 2266
6.538.4 Member Function Documentation . . . 2266
6.539 std::chrono::hh_mm_ss< _Duration > Class Template Reference . . . 2268
6.539.1 Detailed Description . . . 2268
6.540 std::locale::id Class Reference . . . 2268
6.540.1 Detailed Description . . . 2269
6.540.2 Constructor & Destructor Documentation . . . 2269
6.540.3 Friends And Related Symbol Documentation . . . 2269
6.541 std::thread::id Class Reference . . . 2270
6.541.1 Detailed Description . . . 2270
6.542 std::identity Struct Reference . . . 2271
6.542.1 Detailed Description . . . 2271
6.543 std::experimental::fundamentals_v1::in_place_t Struct Reference . . . 2271
6.543.1 Detailed Description . . . 2271
6.544 std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > Class Template Reference 2271
6.544.1 Detailed Description . . . 2272
6.544.2 Member Typedef Documentation . . . 2272
6.544.3 Constructor & Destructor Documentation . . . 2272
6.544.4 Member Function Documentation . . . 2273
6.544.5 Friends And Related Symbol Documentation . . . 2275
6.545 std::indirect_array< _Tp > Class Template Reference . . . 2275
6.545.1 Detailed Description . . . 2276
6.545.2 Member Function Documentation . . . 2277
6.546 std::initializer_list< _E > Class Template Reference . . . 2278
6.546.1 Detailed Description . . . 2278
6.547 std::inout_ptr_t< _Smart, _Pointer, _Args > Class Template Reference . . . 2279
6.547.1 Detailed Description . . . 2279

6.548 std::input_iterator_tag Struct Reference . . . 2279
6.548.1 Detailed Description . . . 2280
6.549 __gnu_pbds::insert_error Struct Reference . . . 2280
6.549.1 Detailed Description . . . 2281
6.549.2 Member Function Documentation . . . 2281
6.550 std::insert_iterator< _Container > Class Template Reference . . . 2281
6.550.1 Detailed Description . . . 2282
6.550.2 Member Typedef Documentation . . . 2283
6.550.3 Constructor & Destructor Documentation . . . 2283
6.550.4 Member Function Documentation . . . 2283
6.551 std::integer_sequence< _Tp, _Idx > Struct Template Reference . . . 2284
6.551.1 Detailed Description . . . 2284
6.552 std::integral_constant< _Tp, __v > Struct Template Reference . . . 2285
6.552.1 Detailed Description . . . 2285
6.553 std::invalid_argument Class Reference . . . 2285
6.553.1 Detailed Description . . . 2286
6.553.2 Member Function Documentation . . . 2286
6.554 std::ios_base Class Reference . . . 2286
6.554.1 Detailed Description . . . 2289
6.554.2 Member Typedef Documentation . . . 2290
6.554.3 Member Enumeration Documentation . . . 2291
6.554.4 Constructor & Destructor Documentation . . . 2292
6.554.5 Member Function Documentation . . . 2292
6.554.6 Member Data Documentation . . . 2297
6.555 std::is_abstract< _Tp > Struct Template Reference . . . 2301
6.555.1 Detailed Description . . . 2302
6.556 std::is_arithmetic< _Tp > Struct Template Reference . . . 2302
6.556.1 Detailed Description . . . 2302
6.557 std::is_array< _Tp > Struct Template Reference . . . 2302
6.557.1 Detailed Description . . . 2302
6.558 std::is_assignable< _Tp, _Up > Struct Template Reference . . . 2302
6.558.1 Detailed Description . . . 2302
6.559 std::is_base_of< _Base, _Derived > Struct Template Reference . . . 2302
6.559.1 Detailed Description . . . 2303
6.560 std::is_bind_expression< _Tp > Struct Template Reference . . . 2303
6.560.1 Detailed Description . . . 2303
6.561 std::is_bind_expression< _Bind< _Signature > > Struct Template Reference . . . 2303
6.561.1 Detailed Description . . . 2303
6.562 std::is_bind_expression< _Bind_result< _Result, _Signature > > Struct Template Reference . . . 2303

6.562.1 Detailed Description . . . 2303
6.563 std::is_bind_expression< const _Bind< _Signature > > Struct Template Reference . . . 2304
6.563.1 Detailed Description . . . 2304
6.564 std::is_bind_expression< const _Bind_result< _Result, _Signature > > Struct Template Reference . . . 2304
6.564.1 Detailed Description . . . 2304
6.565 std::is_bind_expression< const volatile _Bind< _Signature > > Struct Template Reference . . . 2304
6.565.1 Detailed Description . . . 2304
6.566 std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > > Struct Template Reference . . . 2304
6.566.1 Detailed Description . . . 2304
6.567 std::is_bind_expression< volatile _Bind< _Signature > > Struct Template Reference . . . 2305
6.567.1 Detailed Description . . . 2305
6.568 std::is_bind_expression< volatile _Bind_result< _Result, _Signature > > Struct Template Reference . . . 2305
6.568.1 Detailed Description . . . 2305
6.569 std::is_class< _Tp > Struct Template Reference . . . 2305
6.569.1 Detailed Description . . . 2305
6.570 std::is_compound< _Tp > Struct Template Reference . . . 2305
6.570.1 Detailed Description . . . 2305
6.571 std::is_const< _Tp > Struct Template Reference . . . 2306
6.571.1 Detailed Description . . . 2306
6.572 std::is_constructible< _Tp, _Args > Struct Template Reference . . . 2306
6.572.1 Detailed Description . . . 2306
6.573 std::is_copy_assignable< _Tp > Struct Template Reference . . . 2306
6.573.1 Detailed Description . . . 2306
6.574 std::is_copy_constructible< _Tp > Struct Template Reference . . . 2306
6.574.1 Detailed Description . . . 2306
6.575 std::is_default_constructible< _Tp > Struct Template Reference . . . 2306
6.575.1 Detailed Description . . . 2307
6.576 std::is_destructible< _Tp > Struct Template Reference . . . 2307
6.576.1 Detailed Description . . . 2307
6.577 std::is_empty< _Tp > Struct Template Reference . . . 2307
6.577.1 Detailed Description . . . 2307
6.578 std::is_enum< _Tp > Struct Template Reference . . . 2307
6.578.1 Detailed Description . . . 2307
6.579 std::is_error_code_enum< _Tp > Struct Template Reference . . . 2307
6.579.1 Detailed Description . . . 2308
6.580 std::is_error_code_enum< future_errc > Struct Reference . . . 2308
6.580.1 Detailed Description . . . 2308
6.581 std::is_error_condition_enum< _Tp > Struct Template Reference . . . 2308

6.581.1 Detailed Description 2308
6.582 std::is_floating_point< _Tp > Struct Template Reference 2308
6.582.1 Detailed Description 2309
6.583 std::is_function< _Tp > Struct Template Reference 2309
6.583.1 Detailed Description 2309
6.584 std::is_fundamental< _Tp > Struct Template Reference 2309
6.584.1 Detailed Description 2309
6.585 std::is_integral< _Tp > Struct Template Reference 2309
6.585.1 Detailed Description 2309
6.586 std::is_layout_compatible< _Tp, _Up > Struct Template Reference 2309
6.586.1 Detailed Description 2310
6.587 std::is_literal_type< _Tp > Struct Template Reference 2310
6.587.1 Detailed Description 2310
6.588 std::is_lvalue_reference< typename > Struct Template Reference 2310
6.588.1 Detailed Description 2310
6.589 std::is_member_function_pointer< _Tp > Struct Template Reference 2310
6.589.1 Detailed Description 2310
6.590 std::is_member_object_pointer< _Tp > Struct Template Reference 2311
6.590.1 Detailed Description 2311
6.591 std::is_member_pointer< _Tp > Struct Template Reference 2311
6.591.1 Detailed Description 2311
6.592 std::is_move_assignable< _Tp > Struct Template Reference 2311
6.592.1 Detailed Description 2311
6.593 std::is_move_constructible< _Tp > Struct Template Reference 2311
6.593.1 Detailed Description 2311
6.594 std::is_nothrow_assignable< _Tp, _Up > Struct Template Reference 2311
6.594.1 Detailed Description 2312
6.595 std::is_nothrow_constructible< _Tp, _Args > Struct Template Reference 2312
6.595.1 Detailed Description 2312
6.596 std::is_nothrow_copy_assignable< _Tp > Struct Template Reference 2312
6.596.1 Detailed Description 2312
6.597 std::is_nothrow_copy_constructible< _Tp > Struct Template Reference 2312
6.597.1 Detailed Description 2312
6.598 std::is_nothrow_default_constructible< _Tp > Struct Template Reference 2312
6.598.1 Detailed Description 2313
6.599 std::is_nothrow_destructible< _Tp > Struct Template Reference 2313
6.599.1 Detailed Description 2313
6.600 std::is_nothrow_move_assignable< _Tp > Struct Template Reference 2313
6.600.1 Detailed Description 2313

6.601 std::is_nothrow_move_constructible< _Tp > Struct Template Reference 2313
6.601.1 Detailed Description 2313
6.602 std::is_object< _Tp > Struct Template Reference 2313
6.602.1 Detailed Description 2314
6.603 std::is_placeholder< _Tp > Struct Template Reference 2314
6.603.1 Detailed Description 2314
6.604 std::is_placeholder< _Placeholder< _Num > > Struct Template Reference 2315
6.604.1 Detailed Description 2315
6.605 std::is_pod< _Tp > Struct Template Reference 2316
6.605.1 Detailed Description 2316
6.606 std::is_pointer< _Tp > Struct Template Reference 2316
6.606.1 Detailed Description 2316
6.607 std::is_pointer_interconvertible_base_of< _Base, _Derived > Struct Template Reference 2316
6.607.1 Detailed Description 2316
6.608 std::is_polymorphic< _Tp > Struct Template Reference 2316
6.608.1 Detailed Description 2317
6.609 std::is_reference< _Tp > Struct Template Reference 2317
6.609.1 Detailed Description 2317
6.610 std::is_rvalue_reference< typename > Struct Template Reference 2317
6.610.1 Detailed Description 2317
6.611 std::is_same< _Tp, _Up > Struct Template Reference 2317
6.611.1 Detailed Description 2317
6.612 std::is_scalar< _Tp > Struct Template Reference 2317
6.612.1 Detailed Description 2318
6.613 std::is_signed< _Tp > Struct Template Reference 2318
6.613.1 Detailed Description 2318
6.614 std::is_standard_layout< _Tp > Struct Template Reference 2318
6.614.1 Detailed Description 2318
6.615 std::is_trivial< _Tp > Struct Template Reference 2318
6.615.1 Detailed Description 2318
6.616 std::is_trivially_assignable< _Tp, _Up > Struct Template Reference 2318
6.616.1 Detailed Description 2319
6.617 std::is_trivially_constructible< _Tp, _Args > Struct Template Reference 2319
6.617.1 Detailed Description 2319
6.618 std::is_trivially_copy_assignable< _Tp > Struct Template Reference 2319
6.618.1 Detailed Description 2319
6.619 std::is_trivially_copy_constructible< _Tp > Struct Template Reference 2319
6.619.1 Detailed Description 2319
6.620 std::is_trivially_copyable< _Tp > Struct Template Reference 2319

6.620.1 Detailed Description . . . 2320
6.621 std::is_trivially_default_constructible< _Tp > Struct Template Reference . . . 2320
6.621.1 Detailed Description . . . 2320
6.622 std::is_trivially_destructible< _Tp > Struct Template Reference . . . 2320
6.622.1 Detailed Description . . . 2320
6.623 std::is_trivially_move_assignable< _Tp > Struct Template Reference . . . 2320
6.623.1 Detailed Description . . . 2320
6.624 std::is_trivially_move_constructible< _Tp > Struct Template Reference . . . 2320
6.624.1 Detailed Description . . . 2321
6.625 std::is_union< _Tp > Struct Template Reference . . . 2321
6.625.1 Detailed Description . . . 2321
6.626 std::is_unsigned< _Tp > Struct Template Reference . . . 2321
6.626.1 Detailed Description . . . 2321
6.627 std::is_void< _Tp > Struct Template Reference . . . 2321
6.627.1 Detailed Description . . . 2321
6.628 std::is_volatile< _Tp > Struct Template Reference . . . 2321
6.628.1 Detailed Description . . . 2322
6.629 std::istream_iterator< _Tp, _CharT, _Traits, _Dist > Class Template Reference . . . 2322
6.629.1 Detailed Description . . . 2323
6.629.2 Member Typedef Documentation . . . 2323
6.629.3 Constructor & Destructor Documentation . . . 2323
6.629.4 Friends And Related Symbol Documentation . . . 2324
6.630 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference . . . 2324
6.630.1 Detailed Description . . . 2325
6.630.2 Member Typedef Documentation . . . 2326
6.630.3 Constructor & Destructor Documentation . . . 2327
6.630.4 Member Function Documentation . . . 2327
6.631 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator Struct Reference . . . 2328
6.631.1 Detailed Description . . . 2329
6.632 std::experimental::filesystem::v1::path::iterator Class Reference . . . 2329
6.632.1 Detailed Description . . . 2329
6.633 std::filesystem::path::iterator Class Reference . . . 2330
6.633.1 Detailed Description . . . 2330
6.634 std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference . . . 2330
6.634.1 Detailed Description . . . 2331
6.634.2 Member Typedef Documentation . . . 2331
6.635 std::iterator_traits< _Iterator > Struct Template Reference . . . 2332
6.635.1 Detailed Description . . . 2332
6.636 std::iterator_traits< _Tp * > Struct Template Reference . . . 2332

6.636.1 Detailed Description . . . 2333
6.637 __gnu_pbds::join_error Struct Reference . . . 2333
6.637.1 Detailed Description . . . 2333
6.637.2 Member Function Documentation . . . 2334
6.638 __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > Class Template Reference . . . 2334
6.638.1 Detailed Description . . . 2335
6.639 __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc > Class Template Reference . . . 2336
6.639.1 Detailed Description . . . 2336
6.639.2 Member Typedef Documentation . . . 2337
6.639.3 Constructor & Destructor Documentation . . . 2337
6.639.4 Member Function Documentation . . . 2338
6.640 __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc > Struct Template Reference . . . 2339
6.640.1 Detailed Description . . . 2339
6.641 __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > Class Template Reference . . . 2339
6.641.1 Detailed Description . . . 2341
6.641.2 Member Typedef Documentation . . . 2341
6.641.3 Constructor & Destructor Documentation . . . 2342
6.641.4 Member Function Documentation . . . 2342
6.642 std::length_error Class Reference . . . 2343
6.642.1 Detailed Description . . . 2343
6.642.2 Member Function Documentation . . . 2343
6.643 std::less< _Tp > Struct Template Reference . . . 2344
6.643.1 Detailed Description . . . 2344
6.643.2 Member Typedef Documentation . . . 2344
6.644 std::ranges::less Struct Reference . . . 2345
6.644.1 Detailed Description . . . 2345
6.645 std::less< void > Struct Reference . . . 2345
6.645.1 Detailed Description . . . 2346
6.645.2 Member Typedef Documentation . . . 2346
6.646 std::less_equal< _Tp > Struct Template Reference . . . 2346
6.646.1 Detailed Description . . . 2347
6.646.2 Member Typedef Documentation . . . 2347
6.647 std::ranges::less_equal Struct Reference . . . 2347
6.647.1 Detailed Description . . . 2347
6.648 std::less_equal< void > Struct Reference . . . 2347
6.648.1 Detailed Description . . . 2348

6.648.2 Member Typedef Documentation . . . 2348
6.649 __gnu_cxx::limit_condition::limit_adjustor Struct Reference . . . 2349
6.649.1 Detailed Description . . . 2349
6.650 __gnu_cxx::limit_condition Struct Reference . . . 2349
6.650.1 Detailed Description . . . 2349
6.651 std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference . . . 2349
6.651.1 Detailed Description . . . 2350
6.651.2 Member Typedef Documentation . . . 2351
6.651.3 Constructor & Destructor Documentation . . . 2351
6.651.4 Member Function Documentation . . . 2351
6.651.5 Friends And Related Symbol Documentation . . . 2353
6.651.6 Member Data Documentation . . . 2354
6.652 __gnu_pbds::linear_probe_fn< Size_Type > Class Template Reference . . . 2354
6.652.1 Detailed Description . . . 2355
6.652.2 Member Function Documentation . . . 2355
6.653 std::__debug::list< _Tp, _Allocator > Class Template Reference . . . 2355
6.653.1 Detailed Description . . . 2357
6.654 std::list< _Tp, _Alloc > Class Template Reference . . . 2357
6.654.1 Detailed Description . . . 2361
6.654.2 Constructor & Destructor Documentation . . . 2362
6.654.3 Member Function Documentation . . . 2364
6.655 __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc > Class Template Reference . . . 2379
6.655.1 Detailed Description . . . 2379
6.655.2 Constructor & Destructor Documentation . . . 2380
6.656 __gnu_pbds::list_update_tag Struct Reference . . . 2380
6.656.1 Detailed Description . . . 2380
6.657 std::locale Class Reference . . . 2380
6.657.1 Detailed Description . . . 2382
6.657.2 Member Typedef Documentation . . . 2382
6.657.3 Constructor & Destructor Documentation . . . 2382
6.657.4 Member Function Documentation . . . 2385
6.657.5 Friends And Related Symbol Documentation . . . 2387
6.657.6 Member Data Documentation . . . 2388
6.658 std::lock_guard< _Mutex > Class Template Reference . . . 2389
6.658.1 Detailed Description . . . 2390
6.659 std::logic_error Class Reference . . . 2390
6.659.1 Detailed Description . . . 2391
6.659.2 Constructor & Destructor Documentation . . . 2391
6.659.3 Member Function Documentation . . . 2391

6.660 std::logical_and< _Tp > Struct Template Reference . . . 2391
6.660.1 Detailed Description . . . 2392
6.660.2 Member Typedef Documentation . . . 2392
6.661 std::logical_and< void > Struct Reference . . . 2393
6.661.1 Detailed Description . . . 2393
6.661.2 Member Typedef Documentation . . . 2393
6.662 std::logical_not< _Tp > Struct Template Reference . . . 2394
6.662.1 Detailed Description . . . 2394
6.662.2 Member Typedef Documentation . . . 2394
6.663 std::logical_not< void > Struct Reference . . . 2395
6.663.1 Detailed Description . . . 2395
6.663.2 Member Typedef Documentation . . . 2395
6.664 std::logical_or< _Tp > Struct Template Reference . . . 2396
6.664.1 Detailed Description . . . 2396
6.664.2 Member Typedef Documentation . . . 2396
6.665 std::logical_or< void > Struct Reference . . . 2397
6.665.1 Detailed Description . . . 2397
6.665.2 Member Typedef Documentation . . . 2397
6.666 std::lognormal_distribution< _RealType > Class Template Reference . . . 2398
6.666.1 Detailed Description . . . 2399
6.666.2 Member Typedef Documentation . . . 2399
6.666.3 Member Function Documentation . . . 2399
6.666.4 Friends And Related Symbol Documentation . . . 2400
6.667 __gnu_pbds::detail::lu_counter_metadata< Size_Type > Class Template Reference . . . 2401
6.667.1 Detailed Description . . . 2401
6.668 __gnu_pbds::lu_counter_policy< Max_Count, _Alloc > Class Template Reference . . . 2401
6.668.1 Detailed Description . . . 2402
6.668.2 Member Typedef Documentation . . . 2402
6.668.3 Member Enumeration Documentation . . . 2403
6.668.4 Member Function Documentation . . . 2403
6.669 __gnu_pbds::detail::lu_counter_policy_base< Size_Type > Class Template Reference . . . 2403
6.669.1 Detailed Description . . . 2404
6.670 __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > Class Template Reference 2404
6.670.1 Detailed Description . . . 2407
6.671 __gnu_pbds::lu_move_to_front_policy< _Alloc > Class Template Reference . . . 2407
6.671.1 Detailed Description . . . 2407
6.671.2 Member Typedef Documentation . . . 2407
6.671.3 Member Function Documentation . . . 2407
6.672 std::make_signed< _Tp > Struct Template Reference . . . 2408

6.672.1 Detailed Description . . . 2408
6.673 std::make_unsigned< _Tp > Struct Template Reference . . . 2408
6.673.1 Detailed Description . . . 2408
6.674 __gnu_cxx::malloc_allocator< _Tp > Class Template Reference . . . 2408
6.674.1 Detailed Description . . . 2409
6.675 std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference . . . 2409
6.675.1 Detailed Description . . . 2411
6.676 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference . . . 2411
6.676.1 Detailed Description . . . 2414
6.676.2 Constructor & Destructor Documentation . . . 2415
6.676.3 Member Function Documentation . . . 2418
6.677 std::mask_array< _Tp > Class Template Reference . . . 2433
6.677.1 Detailed Description . . . 2434
6.677.2 Member Function Documentation . . . 2435
6.678 __gnu_pbds::detail::mask_based_range_hashing< Size_Type > Class Template Reference . . . 2436
6.678.1 Detailed Description . . . 2437
6.679 std::match_results< _Bi_iter, _Alloc > Class Template Reference . . . 2437
6.679.1 Detailed Description . . . 2439
6.679.2 Constructor & Destructor Documentation . . . 2440
6.679.3 Member Function Documentation . . . 2441
6.680 __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference 2446
6.680.1 Detailed Description . . . 2447
6.681 __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash > Struct Template Reference 2448
6.681.1 Detailed Description . . . 2448
6.682 std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference . . . 2448
6.682.1 Detailed Description . . . 2449
6.682.2 Member Typedef Documentation . . . 2449
6.683 std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference . . . 2450
6.683.1 Detailed Description . . . 2450
6.683.2 Member Typedef Documentation . . . 2450
6.684 std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference . . . 2451
6.684.1 Detailed Description . . . 2451
6.684.2 Member Typedef Documentation . . . 2451
6.685 std::mem_fun_t< _Ret, _Tp > Class Template Reference . . . 2452
6.685.1 Detailed Description . . . 2452
6.685.2 Member Typedef Documentation . . . 2452
6.686 std::pmr::memory_resource Class Reference . . . 2453
6.686.1 Detailed Description . . . 2453

6.687 std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > Class Template Reference 2453
6.687.1 Detailed Description 2454
6.687.2 Member Typedef Documentation 2456
6.687.3 Constructor & Destructor Documentation 2456
6.687.4 Member Function Documentation 2456
6.687.5 Friends And Related Symbol Documentation 2457
6.688 std::messages< _CharT > Class Template Reference 2458
6.688.1 Detailed Description 2460
6.688.2 Member Typedef Documentation 2460
6.688.3 Constructor & Destructor Documentation 2460
6.688.4 Member Function Documentation 2461
6.688.5 Member Data Documentation 2461
6.689 std::messages_base Struct Reference 2462
6.689.1 Detailed Description 2462
6.690 std::messages_byname< _CharT > Class Template Reference 2462
6.690.1 Detailed Description 2464
6.690.2 Member Function Documentation 2464
6.690.3 Member Data Documentation 2464
6.691 std::minus< _Tp > Struct Template Reference 2464
6.691.1 Detailed Description 2465
6.691.2 Member Typedef Documentation 2465
6.692 std::minus< void > Struct Reference 2466
6.692.1 Detailed Description 2466
6.692.2 Member Typedef Documentation 2466
6.693 __gnu_pbds::detail::mod_based_range_hashing< Size_Type > Class Template Reference 2467
6.693.1 Detailed Description 2467
6.694 std::modulus< _Tp > Struct Template Reference 2467
6.694.1 Detailed Description 2468
6.694.2 Member Typedef Documentation 2468
6.695 std::modulus< void > Struct Reference 2468
6.695.1 Detailed Description 2469
6.695.2 Member Typedef Documentation 2469
6.696 std::money_base Class Reference 2469
6.696.1 Detailed Description 2470
6.697 std::money_get< _CharT, _InIter > Class Template Reference 2471
6.697.1 Detailed Description 2472
6.697.2 Member Typedef Documentation 2472
6.697.3 Constructor & Destructor Documentation 2472

6.697.4 Member Function Documentation . . . 2473
6.697.5 Member Data Documentation . . . 2475
6.698 std::money_put< _CharT, _OutIter > Class Template Reference . . . 2475
6.698.1 Detailed Description . . . 2476
6.698.2 Member Typedef Documentation . . . 2476
6.698.3 Constructor & Destructor Documentation . . . 2477
6.698.4 Member Function Documentation . . . 2477
6.698.5 Member Data Documentation . . . 2480
6.699 std::moneypunct< _CharT, _Intl > Class Template Reference . . . 2480
6.699.1 Detailed Description . . . 2482
6.699.2 Member Typedef Documentation . . . 2482
6.699.3 Constructor & Destructor Documentation . . . 2482
6.699.4 Member Function Documentation . . . 2483
6.699.5 Member Data Documentation . . . 2489
6.700 std::moneypunct_byname< _CharT, _Intl > Class Template Reference . . . 2489
6.700.1 Detailed Description . . . 2491
6.700.2 Member Function Documentation . . . 2491
6.700.3 Member Data Documentation . . . 2496
6.701 std::pmr::monotonic_buffer_resource Class Reference . . . 2496
6.701.1 Detailed Description . . . 2497
6.701.2 Member Function Documentation . . . 2498
6.702 std::move_iterator< _Iterator > Class Template Reference . . . 2498
6.702.1 Detailed Description . . . 2499
6.703 std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference . . . 2499
6.703.1 Detailed Description . . . 2500
6.703.2 Constructor & Destructor Documentation . . . 2500
6.703.3 Member Function Documentation . . . 2502
6.703.4 Friends And Related Symbol Documentation . . . 2503
6.704 std::move_sentinel< _Sent > Class Template Reference . . . 2503
6.704.1 Detailed Description . . . 2503
6.705 std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference . . . 2504
6.705.1 Detailed Description . . . 2506
6.706 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference . . . 2506
6.706.1 Detailed Description . . . 2509
6.706.2 Constructor & Destructor Documentation . . . 2509
6.706.3 Member Function Documentation . . . 2512
6.707 std::multiplies< _Tp > Struct Template Reference . . . 2526
6.707.1 Detailed Description . . . 2527

6.707.2 Member Typedef Documentation . . . 2527
6.708 std::multiplies< void > Struct Reference . . . 2527
6.708.1 Detailed Description . . . 2528
6.708.2 Member Typedef Documentation . . . 2528
6.709 std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference . . . 2528
6.709.1 Detailed Description . . . 2530
6.710 std::multiset< _Key, _Compare, _Alloc > Class Template Reference . . . 2530
6.710.1 Detailed Description . . . 2533
6.710.2 Constructor & Destructor Documentation . . . 2533
6.710.3 Member Function Documentation . . . 2536
6.711 __gnu_parallel::multiway_mergesort_exact_tag Struct Reference . . . 2547
6.711.1 Detailed Description . . . 2548
6.711.2 Member Function Documentation . . . 2548
6.712 __gnu_parallel::multiway_mergesort_sampling_tag Struct Reference . . . 2548
6.712.1 Detailed Description . . . 2549
6.712.2 Member Function Documentation . . . 2549
6.713 __gnu_parallel::multiway_mergesort_tag Struct Reference . . . 2550
6.713.1 Detailed Description . . . 2550
6.713.2 Member Function Documentation . . . 2550
6.714 std::mutex Class Reference . . . 2551
6.714.1 Detailed Description . . . 2551
6.715 std::negate< _Tp > Struct Template Reference . . . 2551
6.715.1 Detailed Description . . . 2552
6.715.2 Member Typedef Documentation . . . 2552
6.716 std::negate< void > Struct Reference . . . 2552
6.716.1 Detailed Description . . . 2553
6.716.2 Member Typedef Documentation . . . 2553
6.717 std::negative_binomial_distribution< _IntType > Class Template Reference . . . 2553
6.717.1 Detailed Description . . . 2554
6.717.2 Member Typedef Documentation . . . 2554
6.717.3 Member Function Documentation . . . 2554
6.717.4 Friends And Related Symbol Documentation . . . 2555
6.718 std::nested_exception Class Reference . . . 2556
6.718.1 Detailed Description . . . 2557
6.718.2 Constructor & Destructor Documentation . . . 2557
6.718.3 Member Function Documentation . . . 2557
6.719 __gnu_cxx::limit_condition::never_adjustor Struct Reference . . . 2557
6.719.1 Detailed Description . . . 2557
6.720 __gnu_cxx::random_condition::never_adjustor Struct Reference . . . 2557

6.720.1 Detailed Description . . . 2557
6.721 __gnu_cxx::new_allocator< _Tp > Class Template Reference . . . 2558
6.721.1 Detailed Description . . . 2558
6.722 __gnu_pbds::detail::no_throw_copies< Key, Mapped > Struct Template Reference . . . 2559
6.722.1 Detailed Description . . . 2559
6.723 __gnu_pbds::detail::no_throw_copies< Key, null_type > Struct Template Reference . . . 2559
6.723.1 Detailed Description . . . 2559
6.724 std::normal_distribution< _RealType > Class Template Reference . . . 2559
6.724.1 Detailed Description . . . 2560
6.724.2 Member Typedef Documentation . . . 2561
6.724.3 Constructor & Destructor Documentation . . . 2561
6.724.4 Member Function Documentation . . . 2561
6.724.5 Friends And Related Symbol Documentation . . . 2562
6.725 std::not_equal_to< _Tp > Struct Template Reference . . . 2563
6.725.1 Detailed Description . . . 2564
6.725.2 Member Typedef Documentation . . . 2564
6.726 std::ranges::not_equal_to Struct Reference . . . 2565
6.726.1 Detailed Description . . . 2565
6.727 std::not_equal_to< void > Struct Reference . . . 2565
6.727.1 Detailed Description . . . 2566
6.727.2 Member Typedef Documentation . . . 2566
6.728 __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 > Struct Template Reference . . . 2566
6.728.1 Detailed Description . . . 2566
6.729 __gnu_pbds::null_type Struct Reference . . . 2567
6.729.1 Detailed Description . . . 2567
6.730 std::experimental::fundamentals_v1::nullopt_t Struct Reference . . . 2567
6.730.1 Detailed Description . . . 2568
6.731 std::num_get< _CharT, _InIter > Class Template Reference . . . 2568
6.731.1 Detailed Description . . . 2570
6.731.2 Member Typedef Documentation . . . 2570
6.731.3 Constructor & Destructor Documentation . . . 2570
6.731.4 Member Function Documentation . . . 2570
6.731.5 Member Data Documentation . . . 2585
6.732 std::num_put< _CharT, _OutIter > Class Template Reference . . . 2585
6.732.1 Detailed Description . . . 2587
6.732.2 Member Typedef Documentation . . . 2587
6.732.3 Constructor & Destructor Documentation . . . 2587
6.732.4 Member Function Documentation . . . 2588
6.732.5 Member Data Documentation . . . 2598

6.733 std::numeric_limits< _Tp > Struct Template Reference 2598
6.733.1 Detailed Description 2599
6.733.2 Member Function Documentation 2599
6.733.3 Member Data Documentation 2601
6.734 std::numeric_limits< bool > Struct Reference 2603
6.734.1 Detailed Description 2604
6.734.2 Member Data Documentation 2605
6.735 std::numeric_limits< char > Struct Reference 2607
6.735.1 Detailed Description 2608
6.735.2 Member Data Documentation 2609
6.736 std::numeric_limits< char16_t > Struct Reference 2611
6.736.1 Detailed Description 2612
6.736.2 Member Data Documentation 2613
6.737 std::numeric_limits< char32_t > Struct Reference 2615
6.737.1 Detailed Description 2616
6.737.2 Member Data Documentation 2617
6.738 std::numeric_limits< double > Struct Reference 2619
6.738.1 Detailed Description 2620
6.738.2 Member Data Documentation 2621
6.739 std::numeric_limits< float > Struct Reference 2623
6.739.1 Detailed Description 2624
6.739.2 Member Data Documentation 2625
6.740 std::numeric_limits< int > Struct Reference 2627
6.740.1 Detailed Description 2628
6.740.2 Member Data Documentation 2629
6.741 std::numeric_limits< long > Struct Reference 2631
6.741.1 Detailed Description 2632
6.741.2 Member Data Documentation 2633
6.742 std::numeric_limits< long double > Struct Reference 2635
6.742.1 Detailed Description 2636
6.742.2 Member Data Documentation 2637
6.743 std::numeric_limits< long long > Struct Reference 2639
6.743.1 Detailed Description 2640
6.743.2 Member Data Documentation 2641
6.744 std::numeric_limits< short > Struct Reference 2643
6.744.1 Detailed Description 2644
6.744.2 Member Data Documentation 2645
6.745 std::numeric_limits< signed char > Struct Reference 2647
6.745.1 Detailed Description 2648

6.745.2 Member Data Documentation 2649
6.746 std::numeric_limits< unsigned char > Struct Reference 2651
6.746.1 Detailed Description 2652
6.746.2 Member Data Documentation 2653
6.747 std::numeric_limits< unsigned int > Struct Reference 2655
6.747.1 Detailed Description 2656
6.747.2 Member Data Documentation 2657
6.748 std::numeric_limits< unsigned long > Struct Reference 2659
6.748.1 Detailed Description 2660
6.748.2 Member Data Documentation 2661
6.749 std::numeric_limits< unsigned long long > Struct Reference 2663
6.749.1 Detailed Description 2664
6.749.2 Member Data Documentation 2665
6.750 std::numeric_limits< unsigned short > Struct Reference 2667
6.750.1 Detailed Description 2668
6.750.2 Member Data Documentation 2669
6.751 std::numeric_limits< wchar_t > Struct Reference 2671
6.751.1 Detailed Description 2672
6.751.2 Member Data Documentation 2673
6.752 std::numpunct< _CharT > Class Template Reference 2675
6.752.1 Detailed Description 2676
6.752.2 Member Typedef Documentation 2677
6.752.3 Constructor & Destructor Documentation 2677
6.752.4 Member Function Documentation 2678
6.752.5 Member Data Documentation 2680
6.753 std::numpunct_byname< _CharT > Class Template Reference 2681
6.753.1 Detailed Description 2682
6.753.2 Member Function Documentation 2682
6.753.3 Member Data Documentation 2685
6.754 __gnu_parallel::omp_loop_static_tag Struct Reference 2685
6.754.1 Detailed Description 2685
6.754.2 Member Function Documentation 2685
6.755 __gnu_parallel::omp_loop_tag Struct Reference 2686
6.755.1 Detailed Description 2686
6.755.2 Member Function Documentation 2686
6.756 std::once_flag Struct Reference 2687
6.756.1 Detailed Description 2687
6.756.2 Constructor & Destructor Documentation 2687
6.756.3 Member Function Documentation 2687

6.756.4 Friends And Related Symbol Documentation . . . 2687
6.757 std::experimental::fundamentals_v1::optional< _Tp > Class Template Reference . . . 2688
6.757.1 Detailed Description . . . 2689
6.758 std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference . . . 2689
6.758.1 Detailed Description . . . 2690
6.758.2 Member Typedef Documentation . . . 2690
6.758.3 Constructor & Destructor Documentation . . . 2691
6.758.4 Member Function Documentation . . . 2692
6.759 std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits > Class Template Reference . . . 2692
6.759.1 Detailed Description . . . 2693
6.760 std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference . . . 2693
6.760.1 Detailed Description . . . 2694
6.760.2 Member Typedef Documentation . . . 2694
6.760.3 Constructor & Destructor Documentation . . . 2695
6.760.4 Member Function Documentation . . . 2695
6.761 std::out_of_range Class Reference . . . 2696
6.761.1 Detailed Description . . . 2697
6.761.2 Member Function Documentation . . . 2697
6.762 std::out_ptr_t< _Smart, _Pointer, _Args > Class Template Reference . . . 2697
6.762.1 Detailed Description . . . 2697
6.763 std::output_iterator_tag Struct Reference . . . 2698
6.763.1 Detailed Description . . . 2698
6.764 __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference . . . 2698
6.764.1 Detailed Description . . . 2700
6.764.2 Member Function Documentation . . . 2700
6.765 __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference . . . 2701
6.765.1 Detailed Description . . . 2702
6.765.2 Member Function Documentation . . . 2702
6.766 __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference . . . 2703
6.766.1 Detailed Description . . . 2704
6.766.2 Member Function Documentation . . . 2704
6.767 __gnu_pbds::ov_tree_tag Struct Reference . . . 2704
6.767.1 Detailed Description . . . 2705
6.768 std::overflow_error Class Reference . . . 2705
6.768.1 Detailed Description . . . 2706
6.768.2 Member Function Documentation . . . 2706
6.769 std::owner_less< _Tp > Struct Template Reference . . . 2707

6.769.1 Detailed Description . . . 2707
6.770 std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > > Struct Template Reference . 2707
6.770.1 Detailed Description . . . 2707
6.770.2 Member Typedef Documentation . . . 2707
6.771 std::owner_less< shared_ptr< _Tp > > Struct Template Reference . . . 2708
6.771.1 Detailed Description . . . 2708
6.771.2 Member Typedef Documentation . . . 2708
6.772 std::owner_less< void > Struct Reference . . . 2708
6.772.1 Detailed Description . . . 2709
6.772.2 Member Typedef Documentation . . . 2709
6.773 std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > > Struct Template Reference . . 2709
6.773.1 Detailed Description . . . 2709
6.773.2 Member Typedef Documentation . . . 2709
6.774 std::owner_less< weak_ptr< _Tp > > Struct Template Reference . . . 2710
6.774.1 Detailed Description . . . 2710
6.774.2 Member Typedef Documentation . . . 2710
6.775 std::packaged_task< _Res(_ArgTypes...)> Class Template Reference . . . 2711
6.775.1 Detailed Description . . . 2711
6.776 __gnu_cxx::pair< _T1, _T2 > Struct Template Reference . . . 2711
6.776.1 Detailed Description . . . 2713
6.776.2 Member Typedef Documentation . . . 2713
6.776.3 Constructor & Destructor Documentation . . . 2713
6.776.4 Member Function Documentation . . . 2714
6.776.5 Member Data Documentation . . . 2716
6.777 std::pair< _T1, _T2 > Struct Template Reference . . . 2716
6.777.1 Detailed Description . . . 2718
6.777.2 Member Typedef Documentation . . . 2719
6.777.3 Constructor & Destructor Documentation . . . 2719
6.777.4 Member Function Documentation . . . 2719
6.777.5 Member Data Documentation . . . 2721
6.778 __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference . . . 2722
6.778.1 Detailed Description . . . 2724
6.779 __gnu_pbds::pairing_heap_tag Struct Reference . . . 2724
6.779.1 Detailed Description . . . 2724
6.780 __gnu_parallel::parallel_tag Struct Reference . . . 2724
6.780.1 Detailed Description . . . 2726
6.780.2 Constructor & Destructor Documentation . . . 2726
6.780.3 Member Function Documentation . . . 2726
6.781 std::bernoulli_distribution::param_type Struct Reference . . . 2726

6.781.1 Detailed Description 2727
6.782 std::binomial_distribution< _IntType >::param_type Struct Reference 2727
6.782.1 Detailed Description 2727
6.783 std::cauchy_distribution< _RealType >::param_type Struct Reference 2727
6.783.1 Detailed Description 2728
6.784 std::chi_squared_distribution< _RealType >::param_type Struct Reference 2728
6.784.1 Detailed Description 2728
6.785 std::discrete_distribution< _IntType >::param_type Struct Reference 2728
6.785.1 Detailed Description 2729
6.786 std::exponential_distribution< _RealType >::param_type Struct Reference 2729
6.786.1 Detailed Description 2729
6.787 std::extreme_value_distribution< _RealType >::param_type Struct Reference 2729
6.787.1 Detailed Description 2730
6.788 std::fisher_f_distribution< _RealType >::param_type Struct Reference 2730
6.788.1 Detailed Description 2730
6.789 std::gamma_distribution< _RealType >::param_type Struct Reference 2730
6.789.1 Detailed Description 2731
6.790 std::geometric_distribution< _IntType >::param_type Struct Reference 2731
6.790.1 Detailed Description 2731
6.791 std::lognormal_distribution< _RealType >::param_type Struct Reference 2731
6.791.1 Detailed Description 2732
6.792 std::negative_binomial_distribution< _IntType >::param_type Struct Reference 2732
6.792.1 Detailed Description 2732
6.793 std::normal_distribution< _RealType >::param_type Struct Reference 2732
6.793.1 Detailed Description 2733
6.794 std::piecewise_constant_distribution< _RealType >::param_type Struct Reference 2733
6.794.1 Detailed Description 2734
6.795 std::piecewise_linear_distribution< _RealType >::param_type Struct Reference 2734
6.795.1 Detailed Description 2734
6.796 std::poisson_distribution< _IntType >::param_type Struct Reference 2734
6.796.1 Detailed Description 2735
6.797 std::student_t_distribution< _RealType >::param_type Struct Reference 2735
6.797.1 Detailed Description 2735
6.798 std::uniform_int_distribution< _IntType >::param_type Struct Reference 2735
6.798.1 Detailed Description 2736
6.799 std::uniform_real_distribution< _RealType >::param_type Struct Reference 2736
6.799.1 Detailed Description 2736
6.800 std::weibull_distribution< _RealType >::param_type Struct Reference 2736
6.800.1 Detailed Description 2737

6.801 __gnu_pbds::detail::pat_trie_base Struct Reference . . . 2737
6.801.1 Detailed Description . . . 2738
6.801.2 Member Enumeration Documentation . . . 2738
6.802 __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > Class Template Reference . . . 2738
6.802.1 Detailed Description . . . 2740
6.802.2 Member Enumeration Documentation . . . 2740
6.802.3 Member Function Documentation . . . 2740
6.803 __gnu_pbds::pat_trie_tag Struct Reference . . . 2741
6.803.1 Detailed Description . . . 2742
6.804 std::experimental::filesystem::v1::path Class Reference . . . 2742
6.804.1 Detailed Description . . . 2744
6.805 std::filesystem::path Class Reference . . . 2745
6.805.1 Detailed Description . . . 2747
6.805.2 Member Enumeration Documentation . . . 2747
6.805.3 Friends And Related Symbol Documentation . . . 2748
6.806 std::philox_engine< _UIntType, __w, __n, __r, __consts > Class Template Reference . . . 2748
6.806.1 Detailed Description . . . 2749
6.806.2 Constructor & Destructor Documentation . . . 2750
6.806.3 Member Function Documentation . . . 2750
6.806.4 Friends And Related Symbol Documentation . . . 2751
6.807 std::piecewise_constant_distribution< _RealType > Class Template Reference . . . 2752
6.807.1 Detailed Description . . . 2753
6.807.2 Member Typedef Documentation . . . 2754
6.807.3 Member Function Documentation . . . 2754
6.807.4 Friends And Related Symbol Documentation . . . 2755
6.808 std::piecewise_construct_t Struct Reference . . . 2756
6.808.1 Detailed Description . . . 2756
6.809 std::piecewise_linear_distribution< _RealType > Class Template Reference . . . 2756
6.809.1 Detailed Description . . . 2757
6.809.2 Member Typedef Documentation . . . 2757
6.809.3 Member Function Documentation . . . 2758
6.809.4 Friends And Related Symbol Documentation . . . 2759
6.810 std::plus< _Tp > Struct Template Reference . . . 2760
6.810.1 Detailed Description . . . 2760
6.810.2 Member Typedef Documentation . . . 2760
6.810.3 Member Function Documentation . . . 2761
6.811 __gnu_pbds::point_invalidation_guarantee Struct Reference . . . 2761
6.811.1 Detailed Description . . . 2761

6.812 std::pointer_to_binary_function< _Arg1, _Arg2, _Result > Class Template Reference 2762
6.812.1 Detailed Description 2762
6.812.2 Member Typedef Documentation 2763
6.813 std::pointer_to_unary_function< _Arg, _Result > Class Template Reference 2763
6.813.1 Detailed Description 2764
6.813.2 Member Typedef Documentation 2764
6.814 std::pointer_traits< _Ptr > Struct Template Reference 2764
6.814.1 Detailed Description 2764
6.814.2 Member Typedef Documentation 2765
6.814.3 Member Function Documentation 2765
6.815 std::pointer_traits< _Tp * > Struct Template Reference 2766
6.815.1 Detailed Description 2766
6.815.2 Member Typedef Documentation 2766
6.815.3 Member Function Documentation 2767
6.816 std::poisson_distribution< _IntType > Class Template Reference 2768
6.816.1 Detailed Description 2769
6.816.2 Member Typedef Documentation 2769
6.816.3 Member Function Documentation 2770
6.816.4 Friends And Related Symbol Documentation 2771
6.817 std::pmr::polymorphic_allocator< _Tp > Class Template Reference 2772
6.817.1 Detailed Description 2772
6.818 std::pmr::pool_options Struct Reference 2773
6.818.1 Detailed Description 2773
6.818.2 Member Data Documentation 2773
6.819 __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc > Class Template Reference 2773
6.819.1 Detailed Description 2774
6.819.2 Constructor & Destructor Documentation 2774
6.820 std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference 2775
6.820.1 Detailed Description 2776
6.820.2 Constructor & Destructor Documentation 2777
6.820.3 Member Function Documentation 2779
6.821 __gnu_pbds::priority_queue_tag Struct Reference 2780
6.821.1 Detailed Description 2780
6.822 __gnu_pbds::detail::probe_fn_base< _Alloc > Class Template Reference 2780
6.822.1 Detailed Description 2780
6.823 __gnu_cxx::project1st< _Arg1, _Arg2 > Struct Template Reference 2780
6.823.1 Detailed Description 2781
6.823.2 Member Typedef Documentation 2781
6.824 __gnu_cxx::project2nd< _Arg1, _Arg2 > Struct Template Reference 2781

6.824.1 Detailed Description . . . 2781
6.824.2 Member Typedef Documentation . . . 2781
6.825 std::promise< _Res > Class Template Reference . . . 2782
6.825.1 Detailed Description . . . 2782
6.826 std::promise< _Res & > Class Template Reference . . . 2782
6.826.1 Detailed Description . . . 2783
6.827 std::promise< void > Class Reference . . . 2783
6.827.1 Detailed Description . . . 2784
6.828 std::experimental::fundamentals_v2::propagate_const< _Tp > Class Template Reference . . . 2784
6.828.1 Detailed Description . . . 2785
6.829 __gnu_pbds::quadratic_probe_fn< Size_Type > Class Template Reference . . . 2785
6.829.1 Detailed Description . . . 2785
6.829.2 Member Function Documentation . . . 2785
6.830 std::queue< _Tp, _Sequence > Class Template Reference . . . 2785
6.830.1 Detailed Description . . . 2786
6.830.2 Constructor & Destructor Documentation . . . 2787
6.830.3 Member Function Documentation . . . 2787
6.830.4 Member Data Documentation . . . 2788
6.831 __gnu_parallel::quicksort_tag Struct Reference . . . 2788
6.831.1 Detailed Description . . . 2789
6.831.2 Member Function Documentation . . . 2789
6.832 std::random_access_iterator_tag Struct Reference . . . 2790
6.832.1 Detailed Description . . . 2790
6.833 __gnu_cxx::random_condition Struct Reference . . . 2790
6.833.1 Detailed Description . . . 2791
6.834 std::random_device Class Reference . . . 2791
6.834.1 Detailed Description . . . 2792
6.834.2 Member Typedef Documentation . . . 2792
6.835 std::range_error Class Reference . . . 2792
6.835.1 Detailed Description . . . 2792
6.835.2 Member Function Documentation . . . 2793
6.836 __gnu_pbds::range_invalidation_guarantee Struct Reference . . . 2793
6.836.1 Detailed Description . . . 2793
6.837 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash > Class Template Reference . . . 2794
6.837.1 Detailed Description . . . 2794
6.838 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false > Class Template Reference . . . 2794
6.838.1 Detailed Description . . . 2795

6.839 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true > Class Template Reference . . . 2795
6.839.1 Detailed Description . . . 2795
6.840 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false > Class Template Reference . . . 2795
6.840.1 Detailed Description . . . 2796
6.841 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true > Class Template Reference . . . 2796
6.841.1 Detailed Description . . . 2796
6.842 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash > Class Template Reference . . . 2797
6.842.1 Detailed Description . . . 2797
6.843 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false > Class Template Reference . . . 2797
6.843.1 Detailed Description . . . 2798
6.844 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > Class Template Reference . . . 2798
6.844.1 Detailed Description . . . 2798
6.845 __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false > Class Template Reference . . . 2799
6.845.1 Detailed Description . . . 2799
6.846 std::rank< _Tp > Struct Template Reference . . . 2799
6.846.1 Detailed Description . . . 2800
6.847 std::ratio< _Num, _Den > Struct Template Reference . . . 2800
6.847.1 Detailed Description . . . 2800
6.848 std::ratio_equal< _R1, _R2 > Struct Template Reference . . . 2800
6.848.1 Detailed Description . . . 2801
6.849 std::ratio_greater< _R1, _R2 > Struct Template Reference . . . 2801
6.849.1 Detailed Description . . . 2802
6.850 std::ratio_greater_equal< _R1, _R2 > Struct Template Reference . . . 2802
6.850.1 Detailed Description . . . 2803
6.851 std::ratio_less< _R1, _R2 > Struct Template Reference . . . 2803
6.851.1 Detailed Description . . . 2803
6.852 std::ratio_less_equal< _R1, _R2 > Struct Template Reference . . . 2804
6.852.1 Detailed Description . . . 2804
6.853 std::ratio_not_equal< _R1, _R2 > Struct Template Reference . . . 2804
6.853.1 Detailed Description . . . 2805
6.854 std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference . . . 2805
6.854.1 Detailed Description . . . 2806
6.854.2 Member Typedef Documentation . . . 2806
6.855 __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > Struct Template Reference . . . 2807

6.855.1 Detailed Description . . . 2810
6.856 __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference . . . 2810
6.856.1 Detailed Description . . . 2813
6.856.2 Member Function Documentation . . . 2813
6.857 __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference . . . 2814
6.857.1 Detailed Description . . . 2814
6.858 __gnu_pbds::rb_tree_tag Struct Reference . . . 2814
6.858.1 Detailed Description . . . 2815
6.859 __gnu_pbds::detail::rc< _Node, _Alloc > Class Template Reference . . . 2815
6.859.1 Detailed Description . . . 2816
6.860 __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference . . . 2816
6.860.1 Detailed Description . . . 2818
6.861 __gnu_pbds::rc_binomial_heap_tag Struct Reference . . . 2818
6.861.1 Detailed Description . . . 2818
6.862 std::simd::rebind< _Tp, _Mp, _Traits > Struct Template Reference . . . 2818
6.862.1 Detailed Description . . . 2819
6.863 std::simd::rebind< _Tp, _Vp, _Traits > Struct Template Reference . . . 2819
6.863.1 Detailed Description . . . 2819
6.864 __gnu_pbds::detail::rebind_traits< _Alloc, T > Struct Template Reference . . . 2819
6.864.1 Detailed Description . . . 2819
6.865 std::filesystem::recursive_directory_iterator Class Reference . . . 2819
6.865.1 Detailed Description . . . 2820
6.866 __gnu_cxx::recursive_init_error Class Reference . . . 2821
6.866.1 Detailed Description . . . 2821
6.866.2 Member Function Documentation . . . 2821
6.867 std::recursive_mutex Class Reference . . . 2821
6.867.1 Detailed Description . . . 2822
6.868 std::recursive_timed_mutex Class Reference . . . 2822
6.868.1 Detailed Description . . . 2822
6.869 std::bitset< _Nb >::reference Class Reference . . . 2823
6.869.1 Detailed Description . . . 2823
6.870 std::tr2::dynamic_bitset< _WordT, _Alloc >::reference Class Reference . . . 2823
6.870.1 Detailed Description . . . 2824
6.871 std::reference_wrapper< _Tp > Class Template Reference . . . 2824
6.871.1 Detailed Description . . . 2825
6.871.2 Friends And Related Symbol Documentation . . . 2825
6.872 std::regex_error Class Reference . . . 2825
6.872.1 Detailed Description . . . 2826

6.872.2 Constructor & Destructor Documentation . . . 2826
6.872.3 Member Function Documentation . . . 2827
6.873 std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference . . . 2827
6.873.1 Detailed Description . . . 2828
6.873.2 Constructor & Destructor Documentation . . . 2828
6.873.3 Member Function Documentation . . . 2829
6.874 std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference . . . 2830
6.874.1 Detailed Description . . . 2830
6.874.2 Constructor & Destructor Documentation . . . 2831
6.874.3 Member Function Documentation . . . 2833
6.875 std::regex_traits< _Ch_type > Class Template Reference . . . 2834
6.875.1 Detailed Description . . . 2835
6.875.2 Constructor & Destructor Documentation . . . 2835
6.875.3 Member Function Documentation . . . 2835
6.876 std::remove_all_extents< _Tp > Struct Template Reference . . . 2840
6.876.1 Detailed Description . . . 2840
6.877 std::remove_const< _Tp > Struct Template Reference . . . 2841
6.877.1 Detailed Description . . . 2841
6.878 std::remove_cv< _Tp > Struct Template Reference . . . 2841
6.878.1 Detailed Description . . . 2841
6.879 std::remove_extent< _Tp > Struct Template Reference . . . 2841
6.879.1 Detailed Description . . . 2841
6.880 std::remove_pointer< _Tp > Struct Template Reference . . . 2841
6.880.1 Detailed Description . . . 2842
6.881 std::remove_reference< _Tp > Struct Template Reference . . . 2842
6.881.1 Detailed Description . . . 2842
6.882 std::remove_volatile< _Tp > Struct Template Reference . . . 2842
6.882.1 Detailed Description . . . 2842
6.883 __gnu_pbds::resize_error Struct Reference . . . 2842
6.883.1 Detailed Description . . . 2843
6.883.2 Member Function Documentation . . . 2843
6.884 __gnu_pbds::detail::resize_policy< _Tp > Class Template Reference . . . 2843
6.884.1 Detailed Description . . . 2844
6.885 std::result_of< _Signature > Struct Template Reference . . . 2844
6.885.1 Detailed Description . . . 2844
6.886 std::reverse_iterator< _Iterator > Class Template Reference . . . 2844
6.886.1 Detailed Description . . . 2846
6.886.2 Constructor & Destructor Documentation . . . 2846
6.886.3 Member Function Documentation . . . 2847

6.887 __gnu_cxx::rope< _CharT, _Alloc > Class Template Reference . . . 2849
6.887.1 Detailed Description . . . 2854
6.888 std::runtime_error Class Reference . . . 2854
6.888.1 Detailed Description . . . 2855
6.888.2 Constructor & Destructor Documentation . . . 2855
6.888.3 Member Function Documentation . . . 2855
6.889 __gnu_pbds::sample_probe_fn Class Reference . . . 2855
6.889.1 Detailed Description . . . 2855
6.889.2 Constructor & Destructor Documentation . . . 2856
6.889.3 Member Function Documentation . . . 2856
6.890 __gnu_pbds::sample_range_hashing Class Reference . . . 2856
6.890.1 Detailed Description . . . 2856
6.890.2 Member Typedef Documentation . . . 2857
6.890.3 Constructor & Destructor Documentation . . . 2857
6.890.4 Member Function Documentation . . . 2857
6.891 __gnu_pbds::sample_ranged_hash_fn Class Reference . . . 2857
6.891.1 Detailed Description . . . 2858
6.891.2 Constructor & Destructor Documentation . . . 2858
6.891.3 Member Function Documentation . . . 2858
6.892 __gnu_pbds::sample_ranged_probe_fn Class Reference . . . 2858
6.892.1 Detailed Description . . . 2859
6.893 __gnu_pbds::sample_resize_policy Class Reference . . . 2859
6.893.1 Detailed Description . . . 2860
6.893.2 Member Typedef Documentation . . . 2860
6.893.3 Constructor & Destructor Documentation . . . 2860
6.893.4 Member Function Documentation . . . 2860
6.894 __gnu_pbds::sample_resize_trigger Class Reference . . . 2862
6.894.1 Detailed Description . . . 2862
6.894.2 Member Typedef Documentation . . . 2862
6.894.3 Constructor & Destructor Documentation . . . 2862
6.894.4 Member Function Documentation . . . 2863
6.895 __gnu_pbds::sample_size_policy Class Reference . . . 2864
6.895.1 Detailed Description . . . 2865
6.895.2 Member Typedef Documentation . . . 2865
6.895.3 Constructor & Destructor Documentation . . . 2865
6.895.4 Member Function Documentation . . . 2865
6.896 __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc > Class Template Reference . . . 2866
6.896.1 Detailed Description . . . 2866

6.897 __gnu_pbds::sample_trie_access_traits Struct Reference 2866
6.897.1 Detailed Description 2866
6.897.2 Member Typedef Documentation 2866
6.897.3 Member Function Documentation 2866
6.898 __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference 2867
6.898.1 Detailed Description 2867
6.898.2 Constructor & Destructor Documentation 2867
6.898.3 Member Function Documentation 2867
6.899 __gnu_pbds::sample_update_policy Struct Reference 2868
6.899.1 Detailed Description 2868
6.899.2 Member Typedef Documentation 2868
6.899.3 Constructor & Destructor Documentation 2868
6.899.4 Member Function Documentation 2868
6.900 __gnu_parallel::sampling_tag Struct Reference 2869
6.900.1 Detailed Description 2869
6.900.2 Member Function Documentation 2869
6.901 std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs > Class Template Reference 2870
6.901.1 Detailed Description 2871
6.902 std::seed_seq Class Reference 2871
6.902.1 Detailed Description 2872
6.902.2 Member Typedef Documentation 2872
6.902.3 Constructor & Destructor Documentation 2872
6.903 __gnu_cxx::select1st< _Pair > Struct Template Reference 2872
6.903.1 Detailed Description 2873
6.903.2 Member Typedef Documentation 2873
6.904 __gnu_cxx::select2nd< _Pair > Struct Template Reference 2873
6.904.1 Detailed Description 2873
6.904.2 Member Typedef Documentation 2873
6.905 __gnu_pbds::detail::select_value_type< Key, Mapped > Struct Template Reference 2874
6.905.1 Detailed Description 2874
6.906 __gnu_pbds::detail::select_value_type< Key, null_type > Struct Template Reference 2874
6.906.1 Detailed Description 2874
6.907 std::basic_istream< _CharT, _Traits >::sentry Class Reference 2874
6.907.1 Detailed Description 2875
6.907.2 Member Typedef Documentation 2875
6.907.3 Constructor & Destructor Documentation 2875
6.907.4 Member Function Documentation 2875
6.908 std::basic_ostream< _CharT, _Traits >::sentry Class Reference 2876

6.908.1 Detailed Description . . . 2876
6.908.2 Constructor & Destructor Documentation . . . 2876
6.908.3 Member Function Documentation . . . 2877
6.909 __gnu_pbds::sequence_tag Struct Reference . . . 2877
6.909.1 Detailed Description . . . 2877
6.910 __gnu_parallel::sequential_tag Struct Reference . . . 2877
6.910.1 Detailed Description . . . 2878
6.911 std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference . . . 2878
6.911.1 Detailed Description . . . 2880
6.912 std::set< _Key, _Compare, _Alloc > Class Template Reference . . . 2880
6.912.1 Detailed Description . . . 2882
6.912.2 Member Typedef Documentation . . . 2883
6.912.3 Constructor & Destructor Documentation . . . 2885
6.912.4 Member Function Documentation . . . 2888
6.913 std::shared_future< _Res > Class Template Reference . . . 2899
6.913.1 Detailed Description . . . 2900
6.913.2 Constructor & Destructor Documentation . . . 2900
6.913.3 Member Function Documentation . . . 2901
6.914 std::shared_future< _Res & > Class Template Reference . . . 2901
6.914.1 Detailed Description . . . 2902
6.914.2 Constructor & Destructor Documentation . . . 2902
6.914.3 Member Function Documentation . . . 2902
6.915 std::shared_future< void > Class Reference . . . 2903
6.915.1 Detailed Description . . . 2903
6.915.2 Constructor & Destructor Documentation . . . 2904
6.915.3 Member Function Documentation . . . 2904
6.916 std::shared_lock< _Mutex > Class Template Reference . . . 2904
6.916.1 Detailed Description . . . 2905
6.917 std::shared_ptr< _Tp > Class Template Reference . . . 2905
6.917.1 Detailed Description . . . 2907
6.917.2 Member Typedef Documentation . . . 2908
6.917.3 Constructor & Destructor Documentation . . . 2908
6.917.4 Member Function Documentation . . . 2914
6.918 std::shared_timed_mutex Class Reference . . . 2915
6.918.1 Detailed Description . . . 2915
6.919 std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference . . . 2916
6.919.1 Detailed Description . . . 2916
6.919.2 Member Typedef Documentation . . . 2917
6.919.3 Constructor & Destructor Documentation . . . 2917

6.919.4 Member Function Documentation . . . 2918
6.919.5 Friends And Related Symbol Documentation . . . 2919
6.920 std::ranges::single_view< _Tp > Class Template Reference . . . 2920
6.920.1 Detailed Description . . . 2921
6.921 std::slice Class Reference . . . 2922
6.921.1 Detailed Description . . . 2922
6.921.2 Friends And Related Symbol Documentation . . . 2922
6.922 std::slice_array< _Tp > Class Template Reference . . . 2922
6.922.1 Detailed Description . . . 2923
6.922.2 Member Function Documentation . . . 2924
6.923 __gnu_cxx::slist< _Tp, _Alloc > Class Template Reference . . . 2925
6.923.1 Detailed Description . . . 2927
6.924 std::experimental::filesystem::v1::space_info Struct Reference . . . 2927
6.924.1 Detailed Description . . . 2927
6.925 std::filesystem::space_info Struct Reference . . . 2927
6.925.1 Detailed Description . . . 2927
6.926 __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference . . . 2928
6.926.1 Detailed Description . . . 2930
6.926.2 Member Function Documentation . . . 2931
6.927 __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference . . . 2931
6.927.1 Detailed Description . . . 2932
6.928 __gnu_pbds::splay_tree_tag Struct Reference . . . 2932
6.928.1 Detailed Description . . . 2933
6.929 std::stack< _Tp, _Sequence > Class Template Reference . . . 2933
6.929.1 Detailed Description . . . 2934
6.929.2 Constructor & Destructor Documentation . . . 2935
6.929.3 Member Function Documentation . . . 2935
6.930 __gnu_cxx::stdio_filebuf< _CharT, _Traits > Class Template Reference . . . 2936
6.930.1 Detailed Description . . . 2939
6.930.2 Constructor & Destructor Documentation . . . 2939
6.930.3 Member Function Documentation . . . 2940
6.930.4 Member Data Documentation . . . 2956
6.931 __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits > Class Template Reference . . . 2959
6.931.1 Detailed Description . . . 2961
6.931.2 Member Function Documentation . . . 2961
6.931.3 Member Data Documentation . . . 2974
6.932 std::chrono::steady_clock Struct Reference . . . 2974
6.932.1 Detailed Description . . . 2975

6.933 __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash > Struct Template Reference 2975
6.933.1 Detailed Description 2976
6.934 __gnu_pbds::detail::stored_data< _Tv, _Th, false > Struct Template Reference 2976
6.934.1 Detailed Description 2976
6.935 __gnu_pbds::detail::stored_hash< _Th > Struct Template Reference 2977
6.935.1 Detailed Description 2977
6.936 __gnu_pbds::detail::stored_value< _Tv > Struct Template Reference 2977
6.936.1 Detailed Description 2978
6.937 __gnu_pbds::string_tag Struct Reference 2978
6.937.1 Detailed Description 2979
6.938 std::student_t_distribution< _RealType > Class Template Reference 2979
6.938.1 Detailed Description 2980
6.938.2 Member Typedef Documentation 2980
6.938.3 Member Function Documentation 2980
6.938.4 Friends And Related Symbol Documentation 2981
6.939 std::sub_match< _BiIter > Class Template Reference 2982
6.939.1 Detailed Description 2983
6.939.2 Member Function Documentation 2984
6.940 std::ranges::subrange< _It, _Sent, _Kind > Class Template Reference 2986
6.940.1 Detailed Description 2987
6.941 std::subtract_with_carry_engine< _UIntType, __w, __s, __r > Class Template Reference 2987
6.941.1 Detailed Description 2988
6.941.2 Member Typedef Documentation 2988
6.941.3 Constructor & Destructor Documentation 2988
6.941.4 Member Function Documentation 2989
6.941.5 Friends And Related Symbol Documentation 2990
6.942 __gnu_cxx::subtractive_rng Class Reference 2991
6.942.1 Detailed Description 2992
6.942.2 Member Typedef Documentation 2992
6.942.3 Constructor & Destructor Documentation 2992
6.942.4 Member Function Documentation 2992
6.943 __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits > Struct Template Reference . . 2992
6.943.1 Detailed Description 2993
6.944 std::chrono::system_clock Struct Reference 2993
6.944.1 Detailed Description 2993
6.945 std::system_error Class Reference 2994
6.945.1 Detailed Description 2994
6.945.2 Member Function Documentation 2994
6.946 std::chrono::tai_clock Class Reference 2994

6.946.1 Detailed Description . . . 2995
6.947 __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp > Struct Template Reference . . . 2995
6.947.1 Detailed Description . . . 2996
6.947.2 Constructor & Destructor Documentation . . . 2996
6.947.3 Member Function Documentation . . . 2997
6.948 __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference . . . 2997
6.948.1 Detailed Description . . . 2999
6.949 __gnu_pbds::thin_heap_tag Struct Reference . . . 2999
6.949.1 Detailed Description . . . 2999
6.950 std::thread Class Reference . . . 3000
6.950.1 Detailed Description . . . 3000
6.950.2 Member Function Documentation . . . 3001
6.951 __gnu_cxx::throw_allocator_base< _Tp, _Cond > Class Template Reference . . . 3001
6.951.1 Detailed Description . . . 3002
6.952 __gnu_cxx::throw_allocator_limit< _Tp > Struct Template Reference . . . 3002
6.952.1 Detailed Description . . . 3003
6.953 __gnu_cxx::throw_allocator_random< _Tp > Struct Template Reference . . . 3004
6.953.1 Detailed Description . . . 3005
6.954 __gnu_cxx::throw_value_base< _Cond > Struct Template Reference . . . 3005
6.954.1 Detailed Description . . . 3006
6.955 __gnu_cxx::throw_value_limit Struct Reference . . . 3006
6.955.1 Detailed Description . . . 3007
6.956 __gnu_cxx::throw_value_random Struct Reference . . . 3007
6.956.1 Detailed Description . . . 3008
6.957 std::time_base Class Reference . . . 3008
6.957.1 Detailed Description . . . 3009
6.958 std::time_get< _CharT, _InIter > Class Template Reference . . . 3009
6.958.1 Detailed Description . . . 3011
6.958.2 Member Typedef Documentation . . . 3011
6.958.3 Constructor & Destructor Documentation . . . 3011
6.958.4 Member Function Documentation . . . 3011
6.958.5 Member Data Documentation . . . 3021
6.959 std::time_get_byname< _CharT, _InIter > Class Template Reference . . . 3021
6.959.1 Detailed Description . . . 3022
6.959.2 Member Function Documentation . . . 3023
6.959.3 Member Data Documentation . . . 3032
6.960 std::chrono::time_point< _Clock, _Dur > Class Template Reference . . . 3032
6.960.1 Detailed Description . . . 3033
6.961 std::time_put< _CharT, _OutIter > Class Template Reference . . . 3033

6.961.1 Detailed Description . . . 3034
6.961.2 Member Typedef Documentation . . . 3034
6.961.3 Constructor & Destructor Documentation . . . 3034
6.961.4 Member Function Documentation . . . 3035
6.961.5 Member Data Documentation . . . 3037
6.962 std::time_put_byname< _CharT, _OutIter > Class Template Reference . . . 3037
6.962.1 Detailed Description . . . 3038
6.962.2 Member Function Documentation . . . 3038
6.962.3 Member Data Documentation . . . 3040
6.963 std::timed_mutex Class Reference . . . 3040
6.963.1 Detailed Description . . . 3041
6.964 std::to_chars_result Struct Reference . . . 3041
6.964.1 Detailed Description . . . 3041
6.965 std::chrono::treat_as_floating_point< _Rep > Struct Template Reference . . . 3041
6.965.1 Detailed Description . . . 3042
6.966 __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc > Class Template Reference . 3042
6.966.1 Detailed Description . . . 3043
6.966.2 Member Typedef Documentation . . . 3044
6.966.3 Constructor & Destructor Documentation . . . 3044
6.967 __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp > Struct Template Reference . . . 3045
6.967.1 Detailed Description . . . 3045
6.968 __gnu_pbds::detail::tree_metadata_helper< Node_Update, false > Struct Template Reference . . . 3045
6.968.1 Detailed Description . . . 3045
6.969 __gnu_pbds::detail::tree_metadata_helper< Node_Update, true > Struct Template Reference . . . 3045
6.969.1 Detailed Description . . . 3045
6.970 __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference . . . 3046
6.970.1 Detailed Description . . . 3046
6.971 __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc > Class Template Reference . . . 3046
6.971.1 Detailed Description . . . 3047
6.971.2 Member Function Documentation . . . 3047
6.972 __gnu_pbds::tree_tag Struct Reference . . . 3048
6.972.1 Detailed Description . . . 3048
6.973 __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc > Struct Template Reference . . . 3048
6.973.1 Detailed Description . . . 3048
6.974 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference . . . 3049
6.974.1 Detailed Description . . . 3049

6.974.2 Member Typedef Documentation . . . 3049
6.975 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference . . . 3049
6.975.1 Detailed Description . . . 3050
6.975.2 Member Typedef Documentation . . . 3051
6.976 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference . . . 3051
6.976.1 Detailed Description . . . 3052
6.976.2 Member Typedef Documentation . . . 3052
6.977 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference . . . 3052
6.977.1 Detailed Description . . . 3052
6.977.2 Member Typedef Documentation . . . 3053
6.978 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference . . . 3053
6.978.1 Detailed Description . . . 3054
6.978.2 Member Typedef Documentation . . . 3054
6.979 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference . . . 3054
6.979.1 Detailed Description . . . 3055
6.979.2 Member Typedef Documentation . . . 3055
6.980 __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc > Class Template Reference . . . 3055
6.980.1 Detailed Description . . . 3056
6.980.2 Member Typedef Documentation . . . 3057
6.980.3 Constructor & Destructor Documentation . . . 3057
6.981 __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp > Struct Template Reference . . . 3058
6.981.1 Detailed Description . . . 3058
6.982 __gnu_pbds::detail::trie_metadata_helper< Node_Update, false > Struct Template Reference . . . 3058
6.982.1 Detailed Description . . . 3058
6.983 __gnu_pbds::detail::trie_metadata_helper< Node_Update, true > Struct Template Reference . . . 3058
6.983.1 Detailed Description . . . 3059
6.984 __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference . . . 3059
6.984.1 Detailed Description . . . 3059
6.985 __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference . . . 3059
6.985.1 Detailed Description . . . 3061
6.985.2 Member Function Documentation . . . 3061
6.986 __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference . . . 3062
6.986.1 Detailed Description . . . 3063
6.986.2 Member Function Documentation . . . 3063

6.987 __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference . . . 3063
6.987.1 Detailed Description . . . 3065
6.987.2 Member Typedef Documentation . . . 3065
6.987.3 Member Function Documentation . . . 3065
6.988 __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc > Struct Template Reference . . . 3066
6.988.1 Detailed Description . . . 3067
6.988.2 Member Typedef Documentation . . . 3067
6.988.3 Member Function Documentation . . . 3067
6.989 __gnu_pbds::trie_tag Struct Reference . . . 3068
6.989.1 Detailed Description . . . 3069
6.990 __gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc > Struct Template Reference . . . 3069
6.990.1 Detailed Description . . . 3069
6.991 __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference . . . 3070
6.991.1 Detailed Description . . . 3070
6.991.2 Member Typedef Documentation . . . 3070
6.992 __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference . . . 3071
6.992.1 Detailed Description . . . 3071
6.992.2 Member Typedef Documentation . . . 3071
6.993 __gnu_pbds::trivial_iterator_tag Struct Reference . . . 3072
6.993.1 Detailed Description . . . 3072
6.994 std::try_to_lock_t Struct Reference . . . 3072
6.994.1 Detailed Description . . . 3072
6.995 std::tuple< _Elements > Class Template Reference . . . 3072
6.995.1 Detailed Description . . . 3074
6.996 std::tuple< _T1, _T2 > Class Template Reference . . . 3074
6.996.1 Detailed Description . . . 3076
6.997 std::tuple_element< __i, _Tp > Struct Template Reference . . . 3076
6.997.1 Detailed Description . . . 3076
6.998 std::tuple_element< 0, pair< _Tp1, _Tp2 > > Struct Template Reference . . . 3076
6.998.1 Detailed Description . . . 3076
6.999 std::tuple_element< 1, pair< _Tp1, _Tp2 > > Struct Template Reference . . . 3076
6.999.1 Detailed Description . . . 3076
6.1000 std::tuple_element< __i, tuple< _Types... > > Struct Template Reference . . . 3077
6.1000.1 Detailed Description . . . 3077
6.1001 std::tuple_element< _Ind, array< _Tp, _Nm > > Struct Template Reference . . . 3077

6.1001.1 Detailed Description . . . 3077
6.1002 std::tuple_size< _Tp > Struct Template Reference . . . 3077
6.1002.1 Detailed Description . . . 3077
6.1003 std::tuple_size< array< _Tp, _Nm > > Struct Template Reference . . . 3077
6.1003.1 Detailed Description . . . 3078
6.1004 std::tuple_size< pair< _Tp1, _Tp2 > > Struct Template Reference . . . 3078
6.1004.1 Detailed Description . . . 3079
6.1005 std::tuple_size< tuple< _Elements... > > Struct Template Reference . . . 3079
6.1005.1 Detailed Description . . . 3080
6.1006 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type Struct Reference . . . 3080
6.1006.1 Detailed Description . . . 3081
6.1007 std::type_index Struct Reference . . . 3081
6.1007.1 Detailed Description . . . 3081
6.1008 __gnu_debug::type_info Class Reference . . . 3081
6.1008.1 Detailed Description . . . 3082
6.1008.2 Constructor & Destructor Documentation . . . 3082
6.1008.3 Member Function Documentation . . . 3082
6.1009 std::type_info Class Reference . . . 3082
6.1009.1 Detailed Description . . . 3083
6.1009.2 Constructor & Destructor Documentation . . . 3083
6.1009.3 Member Function Documentation . . . 3083
6.1010 std::type_order< _Tp, _Up > Struct Template Reference . . . 3083
6.1010.1 Detailed Description . . . 3083
6.1011 __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference . . . 3084
6.1011.1 Detailed Description . . . 3085
6.1012 std::chrono::tzdb_list Class Reference . . . 3085
6.1012.1 Detailed Description . . . 3085
6.1012.2 Member Function Documentation . . . 3086
6.1013 __gnu_cxx::unary_compose< _Operation1, _Operation2 > Class Template Reference . . . 3086
6.1013.1 Detailed Description . . . 3087
6.1013.2 Member Typedef Documentation . . . 3087
6.1014 std::unary_function< _Arg, _Result > Struct Template Reference . . . 3088
6.1014.1 Detailed Description . . . 3088
6.1014.2 Member Typedef Documentation . . . 3088
6.1015 std::unary_negate< _Predicate > Class Template Reference . . . 3089
6.1015.1 Detailed Description . . . 3089
6.1015.2 Member Typedef Documentation . . . 3089
6.1016 __gnu_parallel::unbalanced_tag Struct Reference . . . 3090
6.1016.1 Detailed Description . . . 3090

6.1016.2 Member Function Documentation . . . 3090
6.1017 std::underflow_error Class Reference . . . 3091
6.1017.1 Detailed Description . . . 3091
6.1017.2 Member Function Documentation . . . 3091
6.1018 std::underlying_type< _Tp > Struct Template Reference . . . 3092
6.1018.1 Detailed Description . . . 3092
6.1019 std::uniform_int_distribution< _IntType > Class Template Reference . . . 3092
6.1019.1 Detailed Description . . . 3093
6.1019.2 Member Typedef Documentation . . . 3093
6.1019.3 Constructor & Destructor Documentation . . . 3093
6.1019.4 Member Function Documentation . . . 3093
6.1019.5 Friends And Related Symbol Documentation . . . 3094
6.1020 std::uniform_real_distribution< _RealType > Class Template Reference . . . 3094
6.1020.1 Detailed Description . . . 3095
6.1020.2 Member Typedef Documentation . . . 3095
6.1020.3 Constructor & Destructor Documentation . . . 3096
6.1020.4 Member Function Documentation . . . 3096
6.1020.5 Friends And Related Symbol Documentation . . . 3097
6.1021 std::unique_lock< _Mutex > Class Template Reference . . . 3097
6.1021.1 Detailed Description . . . 3098
6.1021.2 Friends And Related Symbol Documentation . . . 3098
6.1022 std::unique_ptr< _Tp, _Dp > Class Template Reference . . . 3098
6.1022.1 Detailed Description . . . 3100
6.1022.2 Constructor & Destructor Documentation . . . 3100
6.1022.3 Member Function Documentation . . . 3102
6.1023 std::unique_ptr< _Tp[], _Dp > Class Template Reference . . . 3104
6.1023.1 Detailed Description . . . 3105
6.1023.2 Constructor & Destructor Documentation . . . 3105
6.1023.3 Member Function Documentation . . . 3107
6.1023.4 Friends And Related Symbol Documentation . . . 3109
6.1024 std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference . . . 3110
6.1024.1 Detailed Description . . . 3112
6.1025 std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference . . . 3112
6.1025.1 Detailed Description . . . 3116
6.1025.2 Member Typedef Documentation . . . 3116
6.1025.3 Constructor & Destructor Documentation . . . 3118
6.1025.4 Member Function Documentation . . . 3120
6.1026 std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference . . . 3139
6.1026.1 Detailed Description . . . 3141

6.1027 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference 3141
6.1027.1 Detailed Description 3145
6.1027.2 Member Typedef Documentation 3145
6.1027.3 Constructor & Destructor Documentation 3147
6.1027.4 Member Function Documentation 3150
6.1028 std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference . . . 3165
6.1028.1 Detailed Description 3167
6.1029 std::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference 3167
6.1029.1 Detailed Description 3170
6.1029.2 Member Typedef Documentation 3170
6.1029.3 Constructor & Destructor Documentation 3172
6.1029.4 Member Function Documentation 3175
6.1030 std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference 3188
6.1030.1 Detailed Description 3191
6.1031 std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference 3191
6.1031.1 Detailed Description 3194
6.1031.2 Member Typedef Documentation 3194
6.1031.3 Constructor & Destructor Documentation 3196
6.1031.4 Member Function Documentation 3199
6.1032 std::pmr::unsynchronized_pool_resource Class Reference 3214
6.1032.1 Detailed Description 3214
6.1032.2 Member Function Documentation 3215
6.1033 std::uses_allocator< typename, typename > Struct Template Reference 3215
6.1033.1 Detailed Description 3215
6.1034 std::uses_allocator< tuple< _Types... >, _Alloc > Struct Template Reference 3215
6.1034.1 Detailed Description 3215
6.1035 std::chrono::utc_clock Class Reference 3215
6.1035.1 Detailed Description 3216
6.1036 std::valarray< _Tp > Class Template Reference 3216
6.1036.1 Detailed Description 3218
6.1036.2 Constructor & Destructor Documentation 3219
6.1036.3 Member Function Documentation 3219
6.1037 std::__debug::vector< _Tp, _Allocator > Class Template Reference 3222
6.1037.1 Detailed Description 3225
6.1037.2 Constructor & Destructor Documentation 3225
6.1038 std::vector< _Tp, _Alloc > Class Template Reference 3225
6.1038.1 Detailed Description 3229
6.1038.2 Constructor & Destructor Documentation 3229
6.1038.3 Member Function Documentation 3232

6.1039 std::vector< bool, _Alloc > Class Template Reference . . . 3245
6.1039.1 Detailed Description . . . 3248
6.1039.2 Member Function Documentation . . . 3249
6.1040 std::ranges::view_base Struct Reference . . . 3249
6.1040.1 Detailed Description . . . 3249
6.1041 std::ranges::view_interface< _Derived > Class Template Reference . . . 3249
6.1041.1 Detailed Description . . . 3250
6.1042 std::wbuffer_convert< _Codecvt, _Elem, _Tr > Class Template Reference . . . 3250
6.1042.1 Detailed Description . . . 3252
6.1042.2 Member Typedef Documentation . . . 3252
6.1042.3 Constructor & Destructor Documentation . . . 3253
6.1042.4 Member Function Documentation . . . 3254
6.1042.5 Member Data Documentation . . . 3266
6.1043 std::weak_ptr< _Tp > Class Template Reference . . . 3267
6.1043.1 Detailed Description . . . 3268
6.1044 std::weibull_distribution< _RealType > Class Template Reference . . . 3268
6.1044.1 Detailed Description . . . 3269
6.1044.2 Member Typedef Documentation . . . 3270
6.1044.3 Member Function Documentation . . . 3270
6.1044.4 Friends And Related Symbol Documentation . . . 3271
6.1045 std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc > Class Template Reference . . . 3271
6.1045.1 Detailed Description . . . 3272
6.1045.2 Constructor & Destructor Documentation . . . 3272
6.1045.3 Member Function Documentation . . . 3273

7 File Documentation **3275**

7.1 algorithm File Reference . . . 3275
7.1.1 Detailed Description . . . 3275
7.2 experimental/algorithm File Reference . . . 3275
7.2.1 Detailed Description . . . 3275
7.3 ext/algorithm File Reference . . . 3275
7.3.1 Detailed Description . . . 3276
7.4 parallel/algorithm File Reference . . . 3277
7.4.1 Detailed Description . . . 3277
7.5 any File Reference . . . 3277
7.5.1 Detailed Description . . . 3277
7.6 experimental/any File Reference . . . 3277
7.6.1 Detailed Description . . . 3278
7.7 array File Reference . . . 3278

7.7.1 Detailed Description . . . 3278
7.8 experimental/array File Reference . . . 3279
7.8.1 Detailed Description . . . 3279
7.9 atomic File Reference . . . 3279
7.9.1 Detailed Description . . . 3283
7.10 auto_ptr.h File Reference . . . 3283
7.10.1 Detailed Description . . . 3283
7.11 backward_warning.h File Reference . . . 3283
7.11.1 Detailed Description . . . 3283
7.12 hash_fun.h File Reference . . . 3283
7.12.1 Detailed Description . . . 3284
7.13 hash_map File Reference . . . 3284
7.13.1 Detailed Description . . . 3284
7.14 hash_set File Reference . . . 3284
7.14.1 Detailed Description . . . 3285
7.15 strstream File Reference . . . 3285
7.15.1 Detailed Description . . . 3285
7.16 barrier File Reference . . . 3285
7.16.1 Detailed Description . . . 3285
7.17 bit File Reference . . . 3286
7.17.1 Detailed Description . . . 3286
7.18 bits/algorithmfwd.h File Reference . . . 3286
7.18.1 Detailed Description . . . 3291
7.19 parallel/algorithmfwd.h File Reference . . . 3291
7.19.1 Detailed Description . . . 3299
7.20 align.h File Reference . . . 3299
7.20.1 Detailed Description . . . 3299
7.21 bits/alloc_traits.h File Reference . . . 3300
7.21.1 Detailed Description . . . 3300
7.22 ext/alloc_traits.h File Reference . . . 3300
7.22.1 Detailed Description . . . 3300
7.23 allocated_ptr.h File Reference . . . 3300
7.23.1 Detailed Description . . . 3300
7.24 allocator.h File Reference . . . 3300
7.24.1 Detailed Description . . . 3301
7.25 atomic_base.h File Reference . . . 3301
7.25.1 Detailed Description . . . 3301
7.26 atomic_futex.h File Reference . . . 3302
7.26.1 Detailed Description . . . 3302

7.27 atomic_lockfree_defines.h File Reference . . . 3302
7.27.1 Detailed Description . . . 3302
7.28 atomic_timed_wait.h File Reference . . . 3302
7.28.1 Detailed Description . . . 3302
7.29 atomic_wait.h File Reference . . . 3302
7.29.1 Detailed Description . . . 3302
7.30 basic_ios.h File Reference . . . 3302
7.30.1 Detailed Description . . . 3303
7.31 basic_ios.tcc File Reference . . . 3303
7.31.1 Detailed Description . . . 3303
7.32 basic_string.h File Reference . . . 3303
7.32.1 Detailed Description . . . 3306
7.33 basic_string.tcc File Reference . . . 3306
7.33.1 Detailed Description . . . 3306
7.34 backward/binders.h File Reference . . . 3306
7.34.1 Detailed Description . . . 3306
7.35 bits/binders.h File Reference . . . 3307
7.35.1 Detailed Description . . . 3307
7.36 boost_concept_check.h File Reference . . . 3307
7.36.1 Detailed Description . . . 3307
7.37 c++0x_warning.h File Reference . . . 3308
7.37.1 Detailed Description . . . 3308
7.38 char_traits.h File Reference . . . 3308
7.38.1 Detailed Description . . . 3308
7.39 charconv.h File Reference . . . 3308
7.39.1 Detailed Description . . . 3308
7.40 chrono.h File Reference . . . 3309
7.40.1 Detailed Description . . . 3312
7.40.2 Function Documentation . . . 3312
7.41 chrono_io.h File Reference . . . 3313
7.41.1 Detailed Description . . . 3316
7.42 codecvt.h File Reference . . . 3316
7.42.1 Detailed Description . . . 3317
7.43 concept_check.h File Reference . . . 3317
7.43.1 Detailed Description . . . 3317
7.44 cow_string.h File Reference . . . 3317
7.44.1 Detailed Description . . . 3318
7.45 cpp_type_traits.h File Reference . . . 3318
7.45.1 Detailed Description . . . 3318

7.46 cpyfunc_impl.h File Reference . . . 3318
7.46.1 Detailed Description . . . 3318
7.47 cxxabi_forced.h File Reference . . . 3318
7.47.1 Detailed Description . . . 3319
7.48 cxxabi_init_exception.h File Reference . . . 3319
7.48.1 Detailed Description . . . 3319
7.49 deque.tcc File Reference . . . 3319
7.49.1 Detailed Description . . . 3321
7.50 enable_special_members.h File Reference . . . 3321
7.50.1 Detailed Description . . . 3321
7.51 erase_if.h File Reference . . . 3321
7.51.1 Detailed Description . . . 3321
7.52 exception.h File Reference . . . 3321
7.52.1 Detailed Description . . . 3321
7.53 exception_defines.h File Reference . . . 3321
7.53.1 Detailed Description . . . 3322
7.54 exception_ptr.h File Reference . . . 3322
7.54.1 Detailed Description . . . 3322
7.54.2 Function Documentation . . . 3322
7.55 formatfwd.h File Reference . . . 3322
7.55.1 Detailed Description . . . 3322
7.56 forward_list.h File Reference . . . 3322
7.56.1 Detailed Description . . . 3323
7.57 forward_list.tcc File Reference . . . 3323
7.57.1 Detailed Description . . . 3323
7.58 bits/fs_dir.h File Reference . . . 3324
7.58.1 Detailed Description . . . 3324
7.59 experimental/bits/fs_dir.h File Reference . . . 3324
7.59.1 Detailed Description . . . 3324
7.60 bits/fs_fwd.h File Reference . . . 3325
7.60.1 Detailed Description . . . 3326
7.61 experimental/bits/fs_fwd.h File Reference . . . 3326
7.61.1 Detailed Description . . . 3328
7.62 bits/fs_ops.h File Reference . . . 3328
7.62.1 Detailed Description . . . 3330
7.63 experimental/bits/fs_ops.h File Reference . . . 3330
7.63.1 Detailed Description . . . 3333
7.64 bits/fs_path.h File Reference . . . 3333
7.64.1 Detailed Description . . . 3333

7.65 experimental/bits/fs_path.h File Reference . . . 3333
7.65.1 Detailed Description . . . 3333
7.66 fstream.tcc File Reference . . . 3334
7.66.1 Detailed Description . . . 3334
7.67 funcref_impl.h File Reference . . . 3334
7.67.1 Detailed Description . . . 3334
7.68 functexcept.h File Reference . . . 3334
7.68.1 Detailed Description . . . 3334
7.69 functional_hash.h File Reference . . . 3334
7.69.1 Detailed Description . . . 3335
7.70 funcwrap.h File Reference . . . 3335
7.70.1 Detailed Description . . . 3336
7.71 gslice.h File Reference . . . 3336
7.71.1 Detailed Description . . . 3336
7.72 gslice_array.h File Reference . . . 3336
7.72.1 Detailed Description . . . 3336
7.73 hash_bytes.h File Reference . . . 3336
7.73.1 Detailed Description . . . 3337
7.74 backward/hashtable.h File Reference . . . 3337
7.74.1 Detailed Description . . . 3337
7.75 bits/hashtable.h File Reference . . . 3337
7.75.1 Detailed Description . . . 3337
7.76 hashtable_policy.h File Reference . . . 3337
7.76.1 Detailed Description . . . 3338
7.77 indirect.h File Reference . . . 3338
7.77.1 Detailed Description . . . 3338
7.78 indirect_array.h File Reference . . . 3338
7.78.1 Detailed Description . . . 3338
7.79 inplace_tags.h File Reference . . . 3338
7.79.1 Detailed Description . . . 3338
7.80 intcmp.h File Reference . . . 3339
7.80.1 Detailed Description . . . 3339
7.81 invoke.h File Reference . . . 3339
7.81.1 Detailed Description . . . 3340
7.82 ios_base.h File Reference . . . 3340
7.82.1 Detailed Description . . . 3341
7.83 iosfwd_file.h File Reference . . . 3341
7.83.1 Detailed Description . . . 3342
7.84 iosfwd_span.h File Reference . . . 3342

7.84.1 Detailed Description . . . 3342
7.85 iosfwd_string.h File Reference . . . 3342
7.85.1 Detailed Description . . . 3342
7.86 iosfwd_sync.h File Reference . . . 3342
7.86.1 Detailed Description . . . 3342
7.87 istream.tcc File Reference . . . 3342
7.87.1 Detailed Description . . . 3343
7.88 iterator_concepts.h File Reference . . . 3343
7.88.1 Detailed Description . . . 3344
7.89 list.tcc File Reference . . . 3344
7.89.1 Detailed Description . . . 3344
7.90 locale_classes.h File Reference . . . 3345
7.90.1 Detailed Description . . . 3345
7.91 locale_classes.tcc File Reference . . . 3345
7.91.1 Detailed Description . . . 3345
7.92 locale_conv.h File Reference . . . 3345
7.92.1 Detailed Description . . . 3346
7.93 locale_facets.h File Reference . . . 3346
7.93.1 Detailed Description . . . 3347
7.94 locale_facets.tcc File Reference . . . 3347
7.94.1 Detailed Description . . . 3348
7.95 locale_facets_nonio.h File Reference . . . 3348
7.95.1 Detailed Description . . . 3349
7.96 locale_facets_nonio.tcc File Reference . . . 3349
7.96.1 Detailed Description . . . 3349
7.97 localefwd.h File Reference . . . 3349
7.97.1 Detailed Description . . . 3350
7.98 mask_array.h File Reference . . . 3350
7.98.1 Detailed Description . . . 3350
7.99 max_size_type.h File Reference . . . 3350
7.99.1 Detailed Description . . . 3350
7.100 memory_resource.h File Reference . . . 3350
7.100.1 Detailed Description . . . 3350
7.101 memoryfwd.h File Reference . . . 3351
7.101.1 Detailed Description . . . 3351
7.102 mofunc_impl.h File Reference . . . 3351
7.102.1 Detailed Description . . . 3351
7.103 monostate.h File Reference . . . 3351
7.103.1 Detailed Description . . . 3351

7.104 move.h File Reference . . . 3351
7.104.1 Detailed Description . . . 3352
7.105 nested_exception.h File Reference . . . 3352
7.105.1 Detailed Description . . . 3353
7.106 bits/new_allocator.h File Reference . . . 3353
7.106.1 Detailed Description . . . 3353
7.107 ext/new_allocator.h File Reference . . . 3353
7.107.1 Detailed Description . . . 3353
7.108 new_except.h File Reference . . . 3353
7.108.1 Detailed Description . . . 3354
7.109 new_throw.h File Reference . . . 3354
7.109.1 Detailed Description . . . 3354
7.110 node_handle.h File Reference . . . 3354
7.110.1 Detailed Description . . . 3354
7.111 optional_ref.h File Reference . . . 3354
7.111.1 Detailed Description . . . 3354
7.112 ostream.h File Reference . . . 3354
7.112.1 Detailed Description . . . 3355
7.113 ostream.tcc File Reference . . . 3356
7.113.1 Detailed Description . . . 3356
7.114 ostream_insert.h File Reference . . . 3356
7.114.1 Detailed Description . . . 3356
7.115 ostream_print.h File Reference . . . 3356
7.115.1 Detailed Description . . . 3356
7.116 out_ptr.h File Reference . . . 3356
7.116.1 Detailed Description . . . 3357
7.117 parse_numbers.h File Reference . . . 3357
7.117.1 Detailed Description . . . 3357
7.118 postypes.h File Reference . . . 3357
7.118.1 Detailed Description . . . 3357
7.119 predefined_ops.h File Reference . . . 3358
7.119.1 Detailed Description . . . 3358
7.120 print.h File Reference . . . 3358
7.120.1 Detailed Description . . . 3358
7.121 ptr_traits.h File Reference . . . 3358
7.121.1 Detailed Description . . . 3359
7.122 quoted_string.h File Reference . . . 3359
7.122.1 Detailed Description . . . 3359
7.123 random.h File Reference . . . 3359

7.123.1 Detailed Description . . . 3362
7.124 bits/random.tcc File Reference . . . 3362
7.124.1 Detailed Description . . . 3366
7.125 ext/random.tcc File Reference . . . 3366
7.125.1 Detailed Description . . . 3368
7.126 range_access.h File Reference . . . 3368
7.126.1 Detailed Description . . . 3369
7.127 ranges_algo.h File Reference . . . 3369
7.127.1 Detailed Description . . . 3374
7.128 ranges_algobase.h File Reference . . . 3374
7.128.1 Detailed Description . . . 3375
7.129 ranges_base.h File Reference . . . 3375
7.129.1 Detailed Description . . . 3377
7.129.2 Variable Documentation . . . 3378
7.130 ranges_cmp.h File Reference . . . 3378
7.130.1 Detailed Description . . . 3378
7.131 ranges_uninitialized.h File Reference . . . 3378
7.131.1 Detailed Description . . . 3378
7.132 ranges_util.h File Reference . . . 3378
7.132.1 Detailed Description . . . 3379
7.133 refwrap.h File Reference . . . 3379
7.133.1 Detailed Description . . . 3380
7.134 regex.h File Reference . . . 3380
7.134.1 Detailed Description . . . 3382
7.135 regex.tcc File Reference . . . 3382
7.135.1 Detailed Description . . . 3382
7.136 regex_automaton.h File Reference . . . 3382
7.136.1 Detailed Description . . . 3383
7.137 regex_automaton.tcc File Reference . . . 3383
7.137.1 Detailed Description . . . 3383
7.138 regex_compiler.h File Reference . . . 3383
7.138.1 Detailed Description . . . 3384
7.139 regex_compiler.tcc File Reference . . . 3384
7.139.1 Detailed Description . . . 3384
7.140 regex_constants.h File Reference . . . 3384
7.140.1 Detailed Description . . . 3385
7.141 regex_error.h File Reference . . . 3386
7.141.1 Detailed Description . . . 3386
7.142 regex_executor.h File Reference . . . 3386

7.142.1 Detailed Description . . . 3386
7.143 regex_executor.tcc File Reference . . . 3387
7.143.1 Detailed Description . . . 3387
7.144 regex_scanner.h File Reference . . . 3387
7.144.1 Detailed Description . . . 3387
7.145 regex_scanner.tcc File Reference . . . 3387
7.145.1 Detailed Description . . . 3387
7.146 requires_hosted.h File Reference . . . 3387
7.146.1 Detailed Description . . . 3387
7.147 sat_arith.h File Reference . . . 3388
7.147.1 Detailed Description . . . 3388
7.148 semaphore_base.h File Reference . . . 3388
7.148.1 Detailed Description . . . 3388
7.149 bits/shared_ptr.h File Reference . . . 3388
7.149.1 Detailed Description . . . 3389
7.150 experimental/bits/shared_ptr.h File Reference . . . 3389
7.150.1 Detailed Description . . . 3391
7.151 shared_ptr_atomic.h File Reference . . . 3391
7.151.1 Detailed Description . . . 3392
7.152 shared_ptr_base.h File Reference . . . 3393
7.152.1 Detailed Description . . . 3394
7.153 slice_array.h File Reference . . . 3394
7.153.1 Detailed Description . . . 3394
7.154 specfun.h File Reference . . . 3394
7.154.1 Detailed Description . . . 3396
7.155 sstream.tcc File Reference . . . 3396
7.155.1 Detailed Description . . . 3396
7.156 std_abs.h File Reference . . . 3396
7.156.1 Detailed Description . . . 3397
7.157 std_function.h File Reference . . . 3397
7.157.1 Detailed Description . . . 3397
7.158 std_mutex.h File Reference . . . 3397
7.158.1 Detailed Description . . . 3398
7.159 std_thread.h File Reference . . . 3398
7.159.1 Detailed Description . . . 3398
7.160 stdexcept_except.h File Reference . . . 3398
7.160.1 Detailed Description . . . 3399
7.161 stdexcept_throw.h File Reference . . . 3399
7.161.1 Detailed Description . . . 3399

7.162 stdexcept_throwdef.h File Reference . . . 3399
7.162.1 Detailed Description . . . 3399
7.163 stdexcept_throwfwd.h File Reference . . . 3399
7.163.1 Detailed Description . . . 3399
7.164 stl_algo.h File Reference . . . 3399
7.164.1 Detailed Description . . . 3406
7.165 stl_algobase.h File Reference . . . 3406
7.165.1 Detailed Description . . . 3411
7.166 stl_bvector.h File Reference . . . 3411
7.166.1 Detailed Description . . . 3412
7.167 stl_construct.h File Reference . . . 3412
7.167.1 Detailed Description . . . 3413
7.168 stl_deque.h File Reference . . . 3413
7.168.1 Detailed Description . . . 3413
7.168.2 Macro Definition Documentation . . . 3413
7.169 stl_function.h File Reference . . . 3414
7.169.1 Detailed Description . . . 3416
7.170 stl_heap.h File Reference . . . 3416
7.170.1 Detailed Description . . . 3417
7.171 bits/stl_iterator.h File Reference . . . 3417
7.171.1 Detailed Description . . . 3420
7.172 debug/stl_iterator.h File Reference . . . 3420
7.172.1 Detailed Description . . . 3421
7.173 stl_iterator_base_funcs.h File Reference . . . 3421
7.173.1 Detailed Description . . . 3422
7.174 stl_iterator_base_types.h File Reference . . . 3422
7.174.1 Detailed Description . . . 3422
7.175 stl_list.h File Reference . . . 3422
7.175.1 Detailed Description . . . 3423
7.176 stl_map.h File Reference . . . 3423
7.176.1 Detailed Description . . . 3424
7.177 stl_multimap.h File Reference . . . 3424
7.177.1 Detailed Description . . . 3425
7.178 stl_multiset.h File Reference . . . 3425
7.178.1 Detailed Description . . . 3426
7.179 stl_numeric.h File Reference . . . 3426
7.179.1 Detailed Description . . . 3426
7.180 stl_pair.h File Reference . . . 3427
7.180.1 Detailed Description . . . 3428

7.181 stl_queue.h File Reference . . . 3428
7.181.1 Detailed Description . . . 3429
7.182 stl_raw_storage_iter.h File Reference . . . 3429
7.182.1 Detailed Description . . . 3429
7.183 stl_relops.h File Reference . . . 3429
7.183.1 Detailed Description . . . 3429
7.184 stl_set.h File Reference . . . 3430
7.184.1 Detailed Description . . . 3430
7.185 stl_stack.h File Reference . . . 3430
7.185.1 Detailed Description . . . 3431
7.186 stl_tempbuf.h File Reference . . . 3431
7.186.1 Detailed Description . . . 3432
7.187 stl_tree.h File Reference . . . 3432
7.187.1 Detailed Description . . . 3432
7.188 stl_uninitialized.h File Reference . . . 3432
7.188.1 Detailed Description . . . 3433
7.189 stl_vector.h File Reference . . . 3433
7.189.1 Detailed Description . . . 3434
7.190 stream_iterator.h File Reference . . . 3434
7.190.1 Detailed Description . . . 3434
7.191 streambuf.tcc File Reference . . . 3434
7.191.1 Detailed Description . . . 3434
7.192 streambuf_iterator.h File Reference . . . 3435
7.192.1 Detailed Description . . . 3435
7.193 bits/string_view.tcc File Reference . . . 3435
7.193.1 Detailed Description . . . 3435
7.194 experimental/bits/string_view.tcc File Reference . . . 3436
7.194.1 Detailed Description . . . 3436
7.195 stringfwd.h File Reference . . . 3436
7.195.1 Detailed Description . . . 3436
7.196 text_encoding-data.h File Reference . . . 3436
7.196.1 Detailed Description . . . 3436
7.197 this_thread_sleep.h File Reference . . . 3436
7.197.1 Detailed Description . . . 3437
7.198 unicode-data.h File Reference . . . 3437
7.198.1 Detailed Description . . . 3437
7.199 unicode.h File Reference . . . 3437
7.199.1 Detailed Description . . . 3438
7.200 uniform_int_dist.h File Reference . . . 3439

7.200.1 Detailed Description . . . 3439
7.201 unique_lock.h File Reference . . . 3439
7.201.1 Detailed Description . . . 3439
7.202 unique_ptr.h File Reference . . . 3439
7.202.1 Detailed Description . . . 3440
7.203 unordered_map.h File Reference . . . 3440
7.203.1 Detailed Description . . . 3442
7.204 unordered_set.h File Reference . . . 3443
7.204.1 Detailed Description . . . 3445
7.205 uses_allocator_args.h File Reference . . . 3445
7.205.1 Detailed Description . . . 3445
7.206 utility.h File Reference . . . 3445
7.206.1 Detailed Description . . . 3446
7.207 valarray_after.h File Reference . . . 3446
7.207.1 Detailed Description . . . 3456
7.208 valarray_array.h File Reference . . . 3456
7.208.1 Detailed Description . . . 3464
7.209 valarray_array.tcc File Reference . . . 3464
7.209.1 Detailed Description . . . 3465
7.210 valarray_before.h File Reference . . . 3465
7.210.1 Detailed Description . . . 3465
7.211 vector.tcc File Reference . . . 3465
7.211.1 Detailed Description . . . 3465
7.212 version.h File Reference . . . 3465
7.212.1 Detailed Description . . . 3470
7.213 bitset File Reference . . . 3470
7.213.1 Detailed Description . . . 3470
7.214 debug/bitset File Reference . . . 3471
7.214.1 Detailed Description . . . 3471
7.215 cassert File Reference . . . 3471
7.215.1 Detailed Description . . . 3471
7.216 ccomplex File Reference . . . 3471
7.216.1 Detailed Description . . . 3471
7.217 tr1/ccomplex File Reference . . . 3472
7.217.1 Detailed Description . . . 3472
7.218 cctype File Reference . . . 3472
7.218.1 Detailed Description . . . 3472
7.219 tr1/cctype File Reference . . . 3472
7.219.1 Detailed Description . . . 3472

7.220 cerrno File Reference . . . 3472
7.220.1 Detailed Description . . . 3472
7.221 cfenv File Reference . . . 3472
7.221.1 Detailed Description . . . 3473
7.222 tr1/cfenv File Reference . . . 3473
7.222.1 Detailed Description . . . 3473
7.223 cfloat File Reference . . . 3473
7.223.1 Detailed Description . . . 3473
7.224 tr1/cfloat File Reference . . . 3473
7.224.1 Detailed Description . . . 3473
7.225 charconv File Reference . . . 3473
7.225.1 Detailed Description . . . 3475
7.226 chrono File Reference . . . 3475
7.226.1 Detailed Description . . . 3477
7.227 experimental/chrono File Reference . . . 3477
7.227.1 Detailed Description . . . 3477
7.228 cinttypes File Reference . . . 3477
7.228.1 Detailed Description . . . 3478
7.229 tr1/cinttypes File Reference . . . 3478
7.229.1 Detailed Description . . . 3478
7.230 ciso646 File Reference . . . 3478
7.230.1 Detailed Description . . . 3478
7.231 climits File Reference . . . 3478
7.231.1 Detailed Description . . . 3478
7.232 tr1/climits File Reference . . . 3478
7.232.1 Detailed Description . . . 3478
7.233 clocale File Reference . . . 3478
7.233.1 Detailed Description . . . 3479
7.234 cmath File Reference . . . 3479
7.234.1 Detailed Description . . . 3485
7.235 ext/cmath File Reference . . . 3485
7.235.1 Detailed Description . . . 3486
7.236 tr1/cmath File Reference . . . 3486
7.236.1 Detailed Description . . . 3488
7.237 codecvt File Reference . . . 3488
7.237.1 Detailed Description . . . 3488
7.238 complex File Reference . . . 3489
7.238.1 Detailed Description . . . 3493
7.239 tr1/complex File Reference . . . 3493

7.239.1 Detailed Description . . . 3494
7.240 complex.h File Reference . . . 3494
7.240.1 Detailed Description . . . 3494
7.241 Concepts . . . 3494
7.242 condition_variable File Reference . . . 3496
7.242.1 Detailed Description . . . 3497
7.243 csetjmp File Reference . . . 3497
7.243.1 Detailed Description . . . 3497
7.244 csignal File Reference . . . 3497
7.244.1 Detailed Description . . . 3497
7.245 cstdalign File Reference . . . 3497
7.245.1 Detailed Description . . . 3498
7.246 cstdarg File Reference . . . 3498
7.246.1 Detailed Description . . . 3498
7.247 tr1/cstdarg File Reference . . . 3498
7.247.1 Detailed Description . . . 3498
7.248 cstdbool File Reference . . . 3498
7.248.1 Detailed Description . . . 3498
7.249 tr1/cstdbool File Reference . . . 3498
7.249.1 Detailed Description . . . 3498
7.250 cstddef File Reference . . . 3499
7.250.1 Detailed Description . . . 3499
7.251 cstdint File Reference . . . 3499
7.251.1 Detailed Description . . . 3499
7.252 tr1/cstdint File Reference . . . 3499
7.252.1 Detailed Description . . . 3499
7.253 cstdio File Reference . . . 3499
7.253.1 Detailed Description . . . 3500
7.254 tr1/cstdio File Reference . . . 3500
7.254.1 Detailed Description . . . 3500
7.255 cstdlib File Reference . . . 3500
7.255.1 Detailed Description . . . 3500
7.256 tr1/cstdlib File Reference . . . 3500
7.256.1 Detailed Description . . . 3500
7.257 cstring File Reference . . . 3500
7.257.1 Detailed Description . . . 3501
7.258 ctgmath File Reference . . . 3501
7.258.1 Detailed Description . . . 3501
7.259 tr1/ctgmath File Reference . . . 3501

7.259.1 Detailed Description . . . 3501
7.260 ctime File Reference . . . 3501
7.260.1 Detailed Description . . . 3502
7.261 tr1/ctime File Reference . . . 3502
7.261.1 Detailed Description . . . 3502
7.262 cuchar File Reference . . . 3502
7.262.1 Detailed Description . . . 3502
7.263 cwchar File Reference . . . 3502
7.263.1 Detailed Description . . . 3502
7.264 tr1/cwchar File Reference . . . 3503
7.264.1 Detailed Description . . . 3503
7.265 cwctype File Reference . . . 3503
7.265.1 Detailed Description . . . 3503
7.266 tr1/cwctype File Reference . . . 3503
7.266.1 Detailed Description . . . 3503
7.267 assertions.h File Reference . . . 3503
7.267.1 Detailed Description . . . 3504
7.268 debug.h File Reference . . . 3504
7.268.1 Detailed Description . . . 3504
7.269 formatter.h File Reference . . . 3504
7.269.1 Detailed Description . . . 3505
7.270 functions.h File Reference . . . 3505
7.270.1 Detailed Description . . . 3507
7.271 helper_functions.h File Reference . . . 3507
7.271.1 Detailed Description . . . 3508
7.272 macros.h File Reference . . . 3508
7.272.1 Detailed Description . . . 3509
7.272.2 Macro Definition Documentation . . . 3509
7.273 map.h File Reference . . . 3511
7.273.1 Detailed Description . . . 3512
7.274 multimap.h File Reference . . . 3512
7.274.1 Detailed Description . . . 3513
7.275 multiset.h File Reference . . . 3513
7.275.1 Detailed Description . . . 3514
7.276 safe_base.h File Reference . . . 3514
7.276.1 Detailed Description . . . 3514
7.277 safe_container.h File Reference . . . 3514
7.277.1 Detailed Description . . . 3514
7.278 safe_iterator.h File Reference . . . 3514

7.278.1 Detailed Description . . . 3515
7.279 safe_iterator.tcc File Reference . . . 3515
7.279.1 Detailed Description . . . 3516
7.280 safe_local_iterator.h File Reference . . . 3516
7.280.1 Detailed Description . . . 3517
7.281 safe_local_iterator.tcc File Reference . . . 3517
7.281.1 Detailed Description . . . 3517
7.282 safe_sequence.h File Reference . . . 3517
7.282.1 Detailed Description . . . 3517
7.283 safe_sequence.tcc File Reference . . . 3517
7.283.1 Detailed Description . . . 3518
7.284 safe_unordered_base.h File Reference . . . 3518
7.284.1 Detailed Description . . . 3518
7.285 safe_unordered_container.h File Reference . . . 3518
7.285.1 Detailed Description . . . 3518
7.286 safe_unordered_container.tcc File Reference . . . 3518
7.286.1 Detailed Description . . . 3518
7.287 set.h File Reference . . . 3518
7.287.1 Detailed Description . . . 3519
7.288 debugging File Reference . . . 3519
7.288.1 Detailed Description . . . 3519
7.289 decimal File Reference . . . 3519
7.289.1 Detailed Description . . . 3528
7.290 debug/deque File Reference . . . 3528
7.290.1 Detailed Description . . . 3529
7.291 deque File Reference . . . 3529
7.291.1 Detailed Description . . . 3529
7.292 experimental/deque File Reference . . . 3529
7.292.1 Detailed Description . . . 3530
7.293 expected File Reference . . . 3530
7.293.1 Detailed Description . . . 3530
7.294 lfts_config.h File Reference . . . 3530
7.294.1 Detailed Description . . . 3530
7.295 ext/numeric_traits.h File Reference . . . 3530
7.295.1 Detailed Description . . . 3531
7.296 propagate_const File Reference . . . 3531
7.296.1 Detailed Description . . . 3532
7.297 simd File Reference . . . 3532
7.297.1 Detailed Description . . . 3533

7.298 aligned_buffer.h File Reference . . . 3533
7.298.1 Detailed Description . . . 3533
7.299 atomicity.h File Reference . . . 3533
7.299.1 Detailed Description . . . 3533
7.300 bitmap_allocator.h File Reference . . . 3533
7.300.1 Detailed Description . . . 3534
7.300.2 Macro Definition Documentation . . . 3534
7.301 cast.h File Reference . . . 3534
7.301.1 Detailed Description . . . 3535
7.302 codecvt_specializations.h File Reference . . . 3535
7.302.1 Detailed Description . . . 3535
7.303 concurrence.h File Reference . . . 3536
7.303.1 Detailed Description . . . 3536
7.304 debug_allocator.h File Reference . . . 3536
7.304.1 Detailed Description . . . 3536
7.305 enc_filebuf.h File Reference . . . 3536
7.305.1 Detailed Description . . . 3536
7.306 extptr_allocator.h File Reference . . . 3537
7.306.1 Detailed Description . . . 3537
7.307 malloc_allocator.h File Reference . . . 3537
7.307.1 Detailed Description . . . 3537
7.308 mt_allocator.h File Reference . . . 3537
7.308.1 Detailed Description . . . 3538
7.309 assoc_container.hpp File Reference . . . 3538
7.309.1 Detailed Description . . . 3538
7.310 bin_search_tree_.hpp File Reference . . . 3539
7.310.1 Detailed Description . . . 3539
7.311 bin_search_tree_/node_iterators.hpp File Reference . . . 3539
7.311.1 Detailed Description . . . 3539
7.312 ov_tree_map_/node_iterators.hpp File Reference . . . 3539
7.312.1 Detailed Description . . . 3540
7.313 point_iterators.hpp File Reference . . . 3540
7.313.1 Detailed Description . . . 3540
7.314 bin_search_tree_/r_erase_fn_imps.hpp File Reference . . . 3540
7.314.1 Detailed Description . . . 3540
7.315 pat_trie_/r_erase_fn_imps.hpp File Reference . . . 3540
7.315.1 Detailed Description . . . 3540
7.316 bin_search_tree_/rotate_fn_imps.hpp File Reference . . . 3540
7.316.1 Detailed Description . . . 3540

7.317 pat_trie_/rotate_fn_imps.hpp File Reference . . . 3540
7.317.1 Detailed Description . . . 3540
7.318 bin_search_tree_/traits.hpp File Reference . . . 3541
7.318.1 Detailed Description . . . 3541
7.319 branch_policy/traits.hpp File Reference . . . 3541
7.319.1 Detailed Description . . . 3541
7.320 ov_tree_map_/traits.hpp File Reference . . . 3541
7.320.1 Detailed Description . . . 3541
7.321 pat_trie_/traits.hpp File Reference . . . 3541
7.321.1 Detailed Description . . . 3541
7.322 rb_tree_map_/traits.hpp File Reference . . . 3542
7.322.1 Detailed Description . . . 3542
7.323 splay_tree_/traits.hpp File Reference . . . 3542
7.323.1 Detailed Description . . . 3542
7.324 binary_heap_.hpp File Reference . . . 3542
7.324.1 Detailed Description . . . 3543
7.325 binary_heap_/const_iterator.hpp File Reference . . . 3543
7.325.1 Detailed Description . . . 3543
7.326 left_child_next_sibling_heap_/const_iterator.hpp File Reference . . . 3543
7.326.1 Detailed Description . . . 3543
7.327 unordered_iterator/const_iterator.hpp File Reference . . . 3543
7.327.1 Detailed Description . . . 3543
7.328 bin_search_tree_/constructors_destructor_fn_imps.hpp File Reference . . . 3544
7.328.1 Detailed Description . . . 3544
7.329 binary_heap_/constructors_destructor_fn_imps.hpp File Reference . . . 3544
7.329.1 Detailed Description . . . 3544
7.330 binomial_heap_/constructors_destructor_fn_imps.hpp File Reference . . . 3544
7.330.1 Detailed Description . . . 3544
7.331 binomial_heap_base_/constructors_destructor_fn_imps.hpp File Reference . . . 3544
7.331.1 Detailed Description . . . 3544
7.332 left_child_next_sibling_heap_/constructors_destructor_fn_imps.hpp File Reference . . . 3544
7.332.1 Detailed Description . . . 3544
7.333 ov_tree_map_/constructors_destructor_fn_imps.hpp File Reference . . . 3544
7.333.1 Detailed Description . . . 3544
7.334 pairing_heap_/constructors_destructor_fn_imps.hpp File Reference . . . 3544
7.334.1 Detailed Description . . . 3544
7.335 pat_trie_/constructors_destructor_fn_imps.hpp File Reference . . . 3544
7.335.1 Detailed Description . . . 3544
7.336 rb_tree_map_/constructors_destructor_fn_imps.hpp File Reference . . . 3544

7.336.1 Detailed Description . . . 3544
7.337 rc_binomial_heap_/constructors_destructor_fn_imps.hpp File Reference . . . 3544
7.337.1 Detailed Description . . . 3544
7.338 splay_tree_/constructors_destructor_fn_imps.hpp File Reference . . . 3545
7.338.1 Detailed Description . . . 3545
7.339 thin_heap_/constructors_destructor_fn_imps.hpp File Reference . . . 3545
7.339.1 Detailed Description . . . 3545
7.340 bin_search_tree_/debug_fn_imps.hpp File Reference . . . 3545
7.340.1 Detailed Description . . . 3545
7.341 binary_heap_/debug_fn_imps.hpp File Reference . . . 3545
7.341.1 Detailed Description . . . 3545
7.342 binomial_heap_/debug_fn_imps.hpp File Reference . . . 3545
7.342.1 Detailed Description . . . 3545
7.343 binomial_heap_base_/debug_fn_imps.hpp File Reference . . . 3545
7.343.1 Detailed Description . . . 3545
7.344 cc_hash_table_map_/debug_fn_imps.hpp File Reference . . . 3545
7.344.1 Detailed Description . . . 3545
7.345 gp_hash_table_map_/debug_fn_imps.hpp File Reference . . . 3545
7.345.1 Detailed Description . . . 3545
7.346 left_child_next_sibling_heap_/debug_fn_imps.hpp File Reference . . . 3545
7.346.1 Detailed Description . . . 3545
7.347 list_update_map_/debug_fn_imps.hpp File Reference . . . 3545
7.347.1 Detailed Description . . . 3545
7.348 ov_tree_map_/debug_fn_imps.hpp File Reference . . . 3546
7.348.1 Detailed Description . . . 3546
7.349 pairing_heap_/debug_fn_imps.hpp File Reference . . . 3546
7.349.1 Detailed Description . . . 3546
7.350 pat_trie_/debug_fn_imps.hpp File Reference . . . 3546
7.350.1 Detailed Description . . . 3546
7.351 rb_tree_map_/debug_fn_imps.hpp File Reference . . . 3546
7.351.1 Detailed Description . . . 3546
7.352 rc_binomial_heap_/debug_fn_imps.hpp File Reference . . . 3546
7.352.1 Detailed Description . . . 3546
7.353 splay_tree_/debug_fn_imps.hpp File Reference . . . 3546
7.353.1 Detailed Description . . . 3546
7.354 thin_heap_/debug_fn_imps.hpp File Reference . . . 3546
7.354.1 Detailed Description . . . 3546
7.355 entry_cmp.hpp File Reference . . . 3546
7.355.1 Detailed Description . . . 3546

7.356 entry_pred.hpp File Reference . . . 3547
7.356.1 Detailed Description . . . 3547
7.357 bin_search_tree_/erase_fn_imps.hpp File Reference . . . 3547
7.357.1 Detailed Description . . . 3547
7.358 binary_heap_/erase_fn_imps.hpp File Reference . . . 3547
7.358.1 Detailed Description . . . 3547
7.359 binomial_heap_base_/erase_fn_imps.hpp File Reference . . . 3547
7.359.1 Detailed Description . . . 3547
7.360 cc_hash_table_map_/erase_fn_imps.hpp File Reference . . . 3547
7.360.1 Detailed Description . . . 3547
7.361 gp_hash_table_map_/erase_fn_imps.hpp File Reference . . . 3547
7.361.1 Detailed Description . . . 3547
7.362 left_child_next_sibling_heap_/erase_fn_imps.hpp File Reference . . . 3547
7.362.1 Detailed Description . . . 3547
7.363 list_update_map_/erase_fn_imps.hpp File Reference . . . 3547
7.363.1 Detailed Description . . . 3547
7.364 ov_tree_map_/erase_fn_imps.hpp File Reference . . . 3548
7.364.1 Detailed Description . . . 3548
7.365 pairing_heap_/erase_fn_imps.hpp File Reference . . . 3548
7.365.1 Detailed Description . . . 3548
7.366 pat_trie_/erase_fn_imps.hpp File Reference . . . 3548
7.366.1 Detailed Description . . . 3548
7.367 rb_tree_map_/erase_fn_imps.hpp File Reference . . . 3548
7.367.1 Detailed Description . . . 3548
7.368 rc_binomial_heap_/erase_fn_imps.hpp File Reference . . . 3548
7.368.1 Detailed Description . . . 3548
7.369 splay_tree_/erase_fn_imps.hpp File Reference . . . 3548
7.369.1 Detailed Description . . . 3548
7.370 thin_heap_/erase_fn_imps.hpp File Reference . . . 3548
7.370.1 Detailed Description . . . 3548
7.371 bin_search_tree_/find_fn_imps.hpp File Reference . . . 3548
7.371.1 Detailed Description . . . 3548
7.372 binary_heap_/find_fn_imps.hpp File Reference . . . 3548
7.372.1 Detailed Description . . . 3548
7.373 binomial_heap_base_/find_fn_imps.hpp File Reference . . . 3548
7.373.1 Detailed Description . . . 3548
7.374 cc_hash_table_map_/find_fn_imps.hpp File Reference . . . 3549
7.374.1 Detailed Description . . . 3549
7.375 gp_hash_table_map_/find_fn_imps.hpp File Reference . . . 3549

7.375.1 Detailed Description . . . 3549
7.376 list_update_map_/find_fn_imps.hpp File Reference . . . 3549
7.376.1 Detailed Description . . . 3549
7.377 pairing_heap_/find_fn_imps.hpp File Reference . . . 3549
7.377.1 Detailed Description . . . 3549
7.378 pat_trie_/find_fn_imps.hpp File Reference . . . 3549
7.378.1 Detailed Description . . . 3549
7.379 rb_tree_map_/find_fn_imps.hpp File Reference . . . 3549
7.379.1 Detailed Description . . . 3549
7.380 splay_tree_/find_fn_imps.hpp File Reference . . . 3549
7.380.1 Detailed Description . . . 3549
7.381 thin_heap_/find_fn_imps.hpp File Reference . . . 3549
7.381.1 Detailed Description . . . 3549
7.382 bin_search_tree_/info_fn_imps.hpp File Reference . . . 3549
7.382.1 Detailed Description . . . 3549
7.383 binary_heap_/info_fn_imps.hpp File Reference . . . 3549
7.383.1 Detailed Description . . . 3549
7.384 cc_hash_table_map_/info_fn_imps.hpp File Reference . . . 3550
7.384.1 Detailed Description . . . 3550
7.385 gp_hash_table_map_/info_fn_imps.hpp File Reference . . . 3550
7.385.1 Detailed Description . . . 3550
7.386 left_child_next_sibling_heap_/info_fn_imps.hpp File Reference . . . 3550
7.386.1 Detailed Description . . . 3550
7.387 list_update_map_/info_fn_imps.hpp File Reference . . . 3550
7.387.1 Detailed Description . . . 3550
7.388 ov_tree_map_/info_fn_imps.hpp File Reference . . . 3550
7.388.1 Detailed Description . . . 3550
7.389 pat_trie_/info_fn_imps.hpp File Reference . . . 3550
7.389.1 Detailed Description . . . 3550
7.390 rb_tree_map_/info_fn_imps.hpp File Reference . . . 3550
7.390.1 Detailed Description . . . 3550
7.391 splay_tree_/info_fn_imps.hpp File Reference . . . 3550
7.391.1 Detailed Description . . . 3550
7.392 bin_search_tree_/insert_fn_imps.hpp File Reference . . . 3550
7.392.1 Detailed Description . . . 3550
7.393 binary_heap_/insert_fn_imps.hpp File Reference . . . 3550
7.393.1 Detailed Description . . . 3550
7.394 binomial_heap_base_/insert_fn_imps.hpp File Reference . . . 3551
7.394.1 Detailed Description . . . 3551

7.395 cc_hash_table_map_/insert_fn_imps.hpp File Reference . . . 3551
7.395.1 Detailed Description . . . 3551
7.396 gp_hash_table_map_/insert_fn_imps.hpp File Reference . . . 3551
7.396.1 Detailed Description . . . 3551
7.397 left_child_next_sibling_heap_/insert_fn_imps.hpp File Reference . . . 3551
7.397.1 Detailed Description . . . 3551
7.398 list_update_map_/insert_fn_imps.hpp File Reference . . . 3551
7.398.1 Detailed Description . . . 3551
7.399 ov_tree_map_/insert_fn_imps.hpp File Reference . . . 3551
7.399.1 Detailed Description . . . 3551
7.400 pairing_heap_/insert_fn_imps.hpp File Reference . . . 3551
7.400.1 Detailed Description . . . 3551
7.401 rb_tree_map_/insert_fn_imps.hpp File Reference . . . 3551
7.401.1 Detailed Description . . . 3551
7.402 rc_binomial_heap_/insert_fn_imps.hpp File Reference . . . 3551
7.402.1 Detailed Description . . . 3551
7.403 splay_tree_/insert_fn_imps.hpp File Reference . . . 3551
7.403.1 Detailed Description . . . 3551
7.404 thin_heap_/insert_fn_imps.hpp File Reference . . . 3552
7.404.1 Detailed Description . . . 3552
7.405 bin_search_tree_/iterators_fn_imps.hpp File Reference . . . 3552
7.405.1 Detailed Description . . . 3552
7.406 binary_heap_/iterators_fn_imps.hpp File Reference . . . 3552
7.406.1 Detailed Description . . . 3552
7.407 cc_hash_table_map_/iterators_fn_imps.hpp File Reference . . . 3552
7.407.1 Detailed Description . . . 3552
7.408 left_child_next_sibling_heap_/iterators_fn_imps.hpp File Reference . . . 3552
7.408.1 Detailed Description . . . 3552
7.409 list_update_map_/iterators_fn_imps.hpp File Reference . . . 3552
7.409.1 Detailed Description . . . 3552
7.410 ov_tree_map_/iterators_fn_imps.hpp File Reference . . . 3552
7.410.1 Detailed Description . . . 3552
7.411 pat_trie_/iterators_fn_imps.hpp File Reference . . . 3552
7.411.1 Detailed Description . . . 3552
7.412 binary_heap_/point_const_iterator.hpp File Reference . . . 3552
7.412.1 Detailed Description . . . 3553
7.413 left_child_next_sibling_heap_/point_const_iterator.hpp File Reference . . . 3553
7.413.1 Detailed Description . . . 3553
7.414 unordered_iterator/point_const_iterator.hpp File Reference . . . 3553

7.414.1 Detailed Description . . . 3553
7.415 bin_search_tree_/policy_access_fn_imps.hpp File Reference . . . 3553
7.415.1 Detailed Description . . . 3553
7.416 binary_heap_/policy_access_fn_imps.hpp File Reference . . . 3553
7.416.1 Detailed Description . . . 3553
7.417 cc_hash_table_map_/policy_access_fn_imps.hpp File Reference . . . 3553
7.417.1 Detailed Description . . . 3553
7.418 gp_hash_table_map_/policy_access_fn_imps.hpp File Reference . . . 3553
7.418.1 Detailed Description . . . 3553
7.419 left_child_next_sibling_heap_/policy_access_fn_imps.hpp File Reference . . . 3554
7.419.1 Detailed Description . . . 3554
7.420 ov_tree_map_/policy_access_fn_imps.hpp File Reference . . . 3554
7.420.1 Detailed Description . . . 3554
7.421 pat_trie_/policy_access_fn_imps.hpp File Reference . . . 3554
7.421.1 Detailed Description . . . 3554
7.422 resize_policy.hpp File Reference . . . 3554
7.422.1 Detailed Description . . . 3554
7.423 bin_search_tree_/split_join_fn_imps.hpp File Reference . . . 3554
7.423.1 Detailed Description . . . 3554
7.424 binary_heap_/split_join_fn_imps.hpp File Reference . . . 3554
7.424.1 Detailed Description . . . 3554
7.425 binomial_heap_base_/split_join_fn_imps.hpp File Reference . . . 3554
7.425.1 Detailed Description . . . 3554
7.426 ov_tree_map_/split_join_fn_imps.hpp File Reference . . . 3555
7.426.1 Detailed Description . . . 3555
7.427 pairing_heap_/split_join_fn_imps.hpp File Reference . . . 3555
7.427.1 Detailed Description . . . 3555
7.428 rb_tree_map_/split_join_fn_imps.hpp File Reference . . . 3555
7.428.1 Detailed Description . . . 3555
7.429 rc_binomial_heap_/split_join_fn_imps.hpp File Reference . . . 3555
7.429.1 Detailed Description . . . 3555
7.430 splay_tree_/split_join_fn_imps.hpp File Reference . . . 3555
7.430.1 Detailed Description . . . 3555
7.431 thin_heap_/split_join_fn_imps.hpp File Reference . . . 3555
7.431.1 Detailed Description . . . 3555
7.432 binary_heap_/trace_fn_imps.hpp File Reference . . . 3555
7.432.1 Detailed Description . . . 3555
7.433 cc_hash_table_map_/trace_fn_imps.hpp File Reference . . . 3555
7.433.1 Detailed Description . . . 3555

7.434 gp_hash_table_map_/trace_fn_imps.hpp File Reference . . . 3555
7.434.1 Detailed Description . . . 3555
7.435 left_child_next_sibling_heap_/trace_fn_imps.hpp File Reference . . . 3555
7.435.1 Detailed Description . . . 3555
7.436 list_update_map_/trace_fn_imps.hpp File Reference . . . 3556
7.436.1 Detailed Description . . . 3556
7.437 pat_trie_/trace_fn_imps.hpp File Reference . . . 3556
7.437.1 Detailed Description . . . 3556
7.438 rc_binomial_heap_/trace_fn_imps.hpp File Reference . . . 3556
7.438.1 Detailed Description . . . 3556
7.439 thin_heap_/trace_fn_imps.hpp File Reference . . . 3556
7.439.1 Detailed Description . . . 3556
7.440 binomial_heap_.hpp File Reference . . . 3556
7.440.1 Detailed Description . . . 3556
7.441 binomial_heap_base_.hpp File Reference . . . 3556
7.441.1 Detailed Description . . . 3557
7.442 branch_policy.hpp File Reference . . . 3557
7.442.1 Detailed Description . . . 3557
7.443 null_node_metadata.hpp File Reference . . . 3557
7.443.1 Detailed Description . . . 3557
7.444 cc_ht_map_.hpp File Reference . . . 3557
7.444.1 Detailed Description . . . 3558
7.445 cmp_fn_imps.hpp File Reference . . . 3558
7.445.1 Detailed Description . . . 3558
7.446 cond_key_dtor_entry_dealtor.hpp File Reference . . . 3558
7.446.1 Detailed Description . . . 3558
7.447 cc_hash_table_map_/constructor_destructor_fn_imps.hpp File Reference . . . 3559
7.447.1 Detailed Description . . . 3559
7.448 gp_hash_table_map_/constructor_destructor_fn_imps.hpp File Reference . . . 3559
7.448.1 Detailed Description . . . 3559
7.449 list_update_map_/constructor_destructor_fn_imps.hpp File Reference . . . 3559
7.450 cc_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp File Reference . . . 3559
7.450.1 Detailed Description . . . 3559
7.451 gp_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp File Reference . . . 3559
7.451.1 Detailed Description . . . 3559
7.452 cc_hash_table_map_/constructor_destructor_store_hash_fn_imps.hpp File Reference . . . 3559
7.452.1 Detailed Description . . . 3559
7.453 gp_hash_table_map_/constructor_destructor_store_hash_fn_imps.hpp File Reference . . . 3559
7.453.1 Detailed Description . . . 3559

7.454 cc_hash_table_map_/debug_no_store_hash_fn_imps.hpp File Reference . . . 3559
7.454.1 Detailed Description . . . 3559
7.455 gp_hash_table_map_/debug_no_store_hash_fn_imps.hpp File Reference . . . 3559
7.455.1 Detailed Description . . . 3559
7.456 cc_hash_table_map_/debug_store_hash_fn_imps.hpp File Reference . . . 3560
7.456.1 Detailed Description . . . 3560
7.457 gp_hash_table_map_/debug_store_hash_fn_imps.hpp File Reference . . . 3560
7.457.1 Detailed Description . . . 3560
7.458 entry_list_fn_imps.hpp File Reference . . . 3560
7.458.1 Detailed Description . . . 3560
7.459 cc_hash_table_map_/erase_no_store_hash_fn_imps.hpp File Reference . . . 3560
7.459.1 Detailed Description . . . 3560
7.460 gp_hash_table_map_/erase_no_store_hash_fn_imps.hpp File Reference . . . 3560
7.460.1 Detailed Description . . . 3560
7.461 cc_hash_table_map_/erase_store_hash_fn_imps.hpp File Reference . . . 3560
7.461.1 Detailed Description . . . 3560
7.462 gp_hash_table_map_/erase_store_hash_fn_imps.hpp File Reference . . . 3560
7.462.1 Detailed Description . . . 3560
7.463 cc_hash_table_map_/find_store_hash_fn_imps.hpp File Reference . . . 3560
7.463.1 Detailed Description . . . 3560
7.464 gp_hash_table_map_/find_store_hash_fn_imps.hpp File Reference . . . 3560
7.464.1 Detailed Description . . . 3560
7.465 cc_hash_table_map_/insert_no_store_hash_fn_imps.hpp File Reference . . . 3560
7.465.1 Detailed Description . . . 3560
7.466 gp_hash_table_map_/insert_no_store_hash_fn_imps.hpp File Reference . . . 3561
7.466.1 Detailed Description . . . 3561
7.467 cc_hash_table_map_/insert_store_hash_fn_imps.hpp File Reference . . . 3561
7.467.1 Detailed Description . . . 3561
7.468 gp_hash_table_map_/insert_store_hash_fn_imps.hpp File Reference . . . 3561
7.468.1 Detailed Description . . . 3561
7.469 cc_hash_table_map_/resize_fn_imps.hpp File Reference . . . 3561
7.469.1 Detailed Description . . . 3561
7.470 gp_hash_table_map_/resize_fn_imps.hpp File Reference . . . 3561
7.470.1 Detailed Description . . . 3561
7.471 cc_hash_table_map_/resize_no_store_hash_fn_imps.hpp File Reference . . . 3561
7.471.1 Detailed Description . . . 3561
7.472 gp_hash_table_map_/resize_no_store_hash_fn_imps.hpp File Reference . . . 3561
7.472.1 Detailed Description . . . 3561
7.473 cc_hash_table_map_/resize_store_hash_fn_imps.hpp File Reference . . . 3561

7.473.1 Detailed Description . . . 3561
7.474 gp_hash_table_map_/resize_store_hash_fn_imps.hpp File Reference . . . 3561
7.474.1 Detailed Description . . . 3561
7.475 size_fn_imps.hpp File Reference . . . 3561
7.475.1 Detailed Description . . . 3561
7.476 cond_dealtor.hpp File Reference . . . 3562
7.476.1 Detailed Description . . . 3562
7.477 container_base_dispatch.hpp File Reference . . . 3562
7.477.1 Detailed Description . . . 3563
7.478 debug_map_base.hpp File Reference . . . 3563
7.478.1 Detailed Description . . . 3563
7.479 eq_by_less.hpp File Reference . . . 3563
7.479.1 Detailed Description . . . 3563
7.480 hash_eq_fn.hpp File Reference . . . 3563
7.480.1 Detailed Description . . . 3563
7.481 find_no_store_hash_fn_imps.hpp File Reference . . . 3563
7.481.1 Detailed Description . . . 3563
7.482 gp_ht_map_.hpp File Reference . . . 3563
7.482.1 Detailed Description . . . 3564
7.483 iterator_fn_imps.hpp File Reference . . . 3564
7.483.1 Detailed Description . . . 3564
7.484 direct_mask_range_hashing_imp.hpp File Reference . . . 3564
7.484.1 Detailed Description . . . 3564
7.485 direct_mod_range_hashing_imp.hpp File Reference . . . 3564
7.485.1 Detailed Description . . . 3564
7.486 linear_probe_fn_imp.hpp File Reference . . . 3564
7.486.1 Detailed Description . . . 3564
7.487 mask_based_range_hashing.hpp File Reference . . . 3565
7.487.1 Detailed Description . . . 3565
7.488 mod_based_range_hashing.hpp File Reference . . . 3565
7.488.1 Detailed Description . . . 3565
7.489 probe_fn_base.hpp File Reference . . . 3565
7.489.1 Detailed Description . . . 3565
7.490 quadratic_probe_fn_imp.hpp File Reference . . . 3565
7.490.1 Detailed Description . . . 3565
7.491 ranged_hash_fn.hpp File Reference . . . 3566
7.491.1 Detailed Description . . . 3566
7.492 ranged_probe_fn.hpp File Reference . . . 3566
7.492.1 Detailed Description . . . 3566

7.493 sample_probe_fn.hpp File Reference . . . 3566
7.493.1 Detailed Description . . . 3567
7.494 sample_range_hashing.hpp File Reference . . . 3567
7.494.1 Detailed Description . . . 3567
7.495 sample_ranged_hash_fn.hpp File Reference . . . 3567
7.495.1 Detailed Description . . . 3567
7.496 sample_ranged_probe_fn.hpp File Reference . . . 3567
7.496.1 Detailed Description . . . 3567
7.497 left_child_next_sibling_heap_.hpp File Reference . . . 3567
7.497.1 Detailed Description . . . 3568
7.498 left_child_next_sibling_heap_/node.hpp File Reference . . . 3568
7.498.1 Detailed Description . . . 3568
7.499 rb_tree_map_/node.hpp File Reference . . . 3568
7.499.1 Detailed Description . . . 3568
7.500 splay_tree_/node.hpp File Reference . . . 3568
7.500.1 Detailed Description . . . 3569
7.501 entry_metadata_base.hpp File Reference . . . 3569
7.501.1 Detailed Description . . . 3569
7.502 lu_map_.hpp File Reference . . . 3569
7.502.1 Detailed Description . . . 3569
7.503 lu_counter_metadata.hpp File Reference . . . 3570
7.503.1 Detailed Description . . . 3570
7.504 sample_update_policy.hpp File Reference . . . 3570
7.504.1 Detailed Description . . . 3570
7.505 ov_tree_map_.hpp File Reference . . . 3570
7.505.1 Detailed Description . . . 3571
7.506 pairing_heap_.hpp File Reference . . . 3571
7.506.1 Detailed Description . . . 3571
7.507 insert_join_fn_imps.hpp File Reference . . . 3571
7.507.1 Detailed Description . . . 3571
7.508 pat_trie_.hpp File Reference . . . 3571
7.508.1 Detailed Description . . . 3572
7.509 pat_trie_base.hpp File Reference . . . 3572
7.509.1 Detailed Description . . . 3572
7.510 split_fn_imps.hpp File Reference . . . 3573
7.510.1 Detailed Description . . . 3573
7.511 synth_access_traits.hpp File Reference . . . 3573
7.511.1 Detailed Description . . . 3573
7.512 update_fn_imps.hpp File Reference . . . 3573

7.512.1 Detailed Description . . . 3573
7.513 priority_queue_base_dispatch.hpp File Reference . . . 3573
7.513.1 Detailed Description . . . 3574
7.514 rb_tree_.hpp File Reference . . . 3574
7.514.1 Detailed Description . . . 3574
7.515 rc.hpp File Reference . . . 3574
7.515.1 Detailed Description . . . 3574
7.516 rc_binomial_heap_.hpp File Reference . . . 3574
7.516.1 Detailed Description . . . 3575
7.517 cc_hash_max_collision_check_resize_trigger_imp.hpp File Reference . . . 3575
7.517.1 Detailed Description . . . 3575
7.518 hash_exponential_size_policy_imp.hpp File Reference . . . 3575
7.518.1 Detailed Description . . . 3575
7.519 hash_load_check_resize_trigger_imp.hpp File Reference . . . 3575
7.519.1 Detailed Description . . . 3575
7.520 hash_load_check_resize_trigger_size_base.hpp File Reference . . . 3575
7.520.1 Detailed Description . . . 3575
7.521 hash_prime_size_policy_imp.hpp File Reference . . . 3575
7.521.1 Detailed Description . . . 3575
7.522 hash_standard_resize_policy_imp.hpp File Reference . . . 3575
7.522.1 Detailed Description . . . 3575
7.523 sample_resize_policy.hpp File Reference . . . 3575
7.523.1 Detailed Description . . . 3576
7.524 sample_resize_trigger.hpp File Reference . . . 3576
7.524.1 Detailed Description . . . 3576
7.525 sample_size_policy.hpp File Reference . . . 3576
7.525.1 Detailed Description . . . 3576
7.526 splay_fn_imps.hpp File Reference . . . 3576
7.526.1 Detailed Description . . . 3576
7.527 splay_tree_.hpp File Reference . . . 3576
7.527.1 Detailed Description . . . 3577
7.528 standard_policies.hpp File Reference . . . 3577
7.528.1 Detailed Description . . . 3577
7.528.2 Enumeration Type Documentation . . . 3577
7.529 thin_heap_.hpp File Reference . . . 3577
7.529.1 Detailed Description . . . 3578
7.530 tree_policy/node_metadata_selector.hpp File Reference . . . 3578
7.530.1 Detailed Description . . . 3578
7.531 trie_policy/node_metadata_selector.hpp File Reference . . . 3578

7.531.1 Detailed Description . . . 3578
7.532 tree_policy/order_statistics_imp.hpp File Reference . . . 3578
7.532.1 Detailed Description . . . 3578
7.533 trie_policy/order_statistics_imp.hpp File Reference . . . 3579
7.533.1 Detailed Description . . . 3579
7.534 sample_tree_node_update.hpp File Reference . . . 3579
7.534.1 Detailed Description . . . 3579
7.535 tree_trace_base.hpp File Reference . . . 3579
7.535.1 Detailed Description . . . 3579
7.536 prefix_search_node_update_imp.hpp File Reference . . . 3579
7.536.1 Detailed Description . . . 3579
7.537 sample_trie_access_traits.hpp File Reference . . . 3579
7.537.1 Detailed Description . . . 3579
7.538 sample_trie_node_update.hpp File Reference . . . 3579
7.538.1 Detailed Description . . . 3580
7.539 trie_policy_base.hpp File Reference . . . 3580
7.539.1 Detailed Description . . . 3580
7.540 trie_string_access_traits_imp.hpp File Reference . . . 3580
7.540.1 Detailed Description . . . 3580
7.541 type_utils.hpp File Reference . . . 3580
7.541.1 Detailed Description . . . 3580
7.542 types_traits.hpp File Reference . . . 3580
7.542.1 Detailed Description . . . 3581
7.543 iterator.hpp File Reference . . . 3581
7.543.1 Detailed Description . . . 3581
7.544 point_iterator.hpp File Reference . . . 3581
7.544.1 Detailed Description . . . 3581
7.545 exception.hpp File Reference . . . 3581
7.545.1 Detailed Description . . . 3582
7.546 hash_policy.hpp File Reference . . . 3582
7.546.1 Detailed Description . . . 3583
7.547 list_update_policy.hpp File Reference . . . 3583
7.547.1 Detailed Description . . . 3583
7.548 priority_queue.hpp File Reference . . . 3583
7.548.1 Detailed Description . . . 3583
7.549 tag_and_trait.hpp File Reference . . . 3583
7.549.1 Detailed Description . . . 3584
7.550 tree_policy.hpp File Reference . . . 3584
7.550.1 Detailed Description . . . 3584

7.551 trie_policy.hpp File Reference . . . 3585
7.551.1 Detailed Description . . . 3585
7.552 pod_char_traits.h File Reference . . . 3585
7.552.1 Detailed Description . . . 3585
7.553 pointer.h File Reference . . . 3586
7.553.1 Detailed Description . . . 3587
7.554 pool_allocator.h File Reference . . . 3587
7.554.1 Detailed Description . . . 3588
7.555 rb_tree File Reference . . . 3588
7.555.1 Detailed Description . . . 3588
7.556 rc_string_base.h File Reference . . . 3588
7.556.1 Detailed Description . . . 3588
7.557 rope File Reference . . . 3589
7.557.1 Detailed Description . . . 3592
7.558 ropeimpl.h File Reference . . . 3592
7.558.1 Detailed Description . . . 3598
7.559 slist File Reference . . . 3598
7.559.1 Detailed Description . . . 3598
7.560 sso_string_base.h File Reference . . . 3599
7.560.1 Detailed Description . . . 3599
7.561 stdio_filebuf.h File Reference . . . 3599
7.561.1 Detailed Description . . . 3599
7.562 stdio_sync_filebuf.h File Reference . . . 3599
7.562.1 Detailed Description . . . 3599
7.563 string_conversions.h File Reference . . . 3599
7.563.1 Detailed Description . . . 3599
7.564 throw_allocator.h File Reference . . . 3600
7.564.1 Detailed Description . . . 3601
7.565 type_traits.h File Reference . . . 3601
7.565.1 Detailed Description . . . 3601
7.566 typelist.h File Reference . . . 3601
7.566.1 Detailed Description . . . 3602
7.567 vstring.h File Reference . . . 3602
7.567.1 Detailed Description . . . 3605
7.568 vstring.tcc File Reference . . . 3605
7.568.1 Detailed Description . . . 3606
7.569 vstring_fwd.h File Reference . . . 3606
7.569.1 Detailed Description . . . 3606
7.570 vstring_util.h File Reference . . . 3606

7.570.1 Detailed Description . . . 3606
7.571 fenv.h File Reference . . . 3607
7.571.1 Detailed Description . . . 3607
7.572 experimental/filesystem File Reference . . . 3607
7.572.1 Detailed Description . . . 3607
7.573 filesystem File Reference . . . 3607
7.573.1 Detailed Description . . . 3607
7.574 flat_map File Reference . . . 3607
7.574.1 Detailed Description . . . 3607
7.575 flat_set File Reference . . . 3607
7.575.1 Detailed Description . . . 3607
7.576 format File Reference . . . 3607
7.576.1 Detailed Description . . . 3607
7.577 debug/forward_list File Reference . . . 3607
7.577.1 Detailed Description . . . 3608
7.578 experimental/forward_list File Reference . . . 3608
7.578.1 Detailed Description . . . 3608
7.579 forward_list File Reference . . . 3609
7.579.1 Detailed Description . . . 3609
7.580 fstream File Reference . . . 3609
7.580.1 Detailed Description . . . 3609
7.581 experimental/functional File Reference . . . 3610
7.581.1 Detailed Description . . . 3610
7.582 ext/functional File Reference . . . 3610
7.582.1 Detailed Description . . . 3611
7.583 functional File Reference . . . 3611
7.583.1 Detailed Description . . . 3613
7.584 future File Reference . . . 3613
7.584.1 Detailed Description . . . 3614
7.585 generator File Reference . . . 3614
7.585.1 Detailed Description . . . 3614
7.586 inplace_vector File Reference . . . 3614
7.586.1 Detailed Description . . . 3614
7.587 iomanip File Reference . . . 3615
7.587.1 Detailed Description . . . 3616
7.588 ios File Reference . . . 3616
7.588.1 Detailed Description . . . 3616
7.589 iosfwd File Reference . . . 3617
7.589.1 Detailed Description . . . 3617

7.590 iostream File Reference . . . 3617
7.590.1 Detailed Description . . . 3617
7.591 istream File Reference . . . 3617
7.591.1 Detailed Description . . . 3618
7.592 experimental/iterator File Reference . . . 3618
7.592.1 Detailed Description . . . 3619
7.593 ext/iterator File Reference . . . 3619
7.593.1 Detailed Description . . . 3619
7.594 iterator File Reference . . . 3619
7.594.1 Detailed Description . . . 3619
7.595 latch File Reference . . . 3619
7.595.1 Detailed Description . . . 3619
7.596 limits File Reference . . . 3619
7.596.1 Detailed Description . . . 3621
7.597 debug/list File Reference . . . 3621
7.597.1 Detailed Description . . . 3621
7.598 experimental/list File Reference . . . 3621
7.598.1 Detailed Description . . . 3622
7.599 list File Reference . . . 3622
7.599.1 Detailed Description . . . 3622
7.600 locale File Reference . . . 3622
7.600.1 Detailed Description . . . 3622
7.601 debug/map File Reference . . . 3622
7.601.1 Detailed Description . . . 3622
7.602 experimental/map File Reference . . . 3622
7.602.1 Detailed Description . . . 3623
7.603 map File Reference . . . 3623
7.603.1 Detailed Description . . . 3623
7.604 math.h File Reference . . . 3623
7.604.1 Detailed Description . . . 3625
7.604.2 Function Documentation . . . 3625
7.605 mdspan File Reference . . . 3627
7.605.1 Detailed Description . . . 3631
7.606 experimental/memory File Reference . . . 3631
7.606.1 Detailed Description . . . 3632
7.607 ext/memory File Reference . . . 3632
7.607.1 Detailed Description . . . 3633
7.608 memory File Reference . . . 3633
7.608.1 Detailed Description . . . 3633

7.609 experimental/memory_resource File Reference . . . 3633
7.609.1 Detailed Description . . . 3633
7.609.2 Function Documentation . . . 3634
7.610 memory_resource File Reference . . . 3634
7.610.1 Detailed Description . . . 3634
7.610.2 Function Documentation . . . 3634
7.611 mutex File Reference . . . 3635
7.611.1 Detailed Description . . . 3635
7.612 numbers File Reference . . . 3635
7.612.1 Detailed Description . . . 3635
7.613 experimental/numeric File Reference . . . 3635
7.613.1 Detailed Description . . . 3635
7.614 ext/numeric File Reference . . . 3636
7.614.1 Detailed Description . . . 3636
7.615 numeric File Reference . . . 3636
7.615.1 Detailed Description . . . 3637
7.616 parallel/numeric File Reference . . . 3637
7.616.1 Detailed Description . . . 3639
7.617 experimental/optional File Reference . . . 3639
7.617.1 Detailed Description . . . 3640
7.618 optional File Reference . . . 3640
7.618.1 Detailed Description . . . 3640
7.619 ostream File Reference . . . 3640
7.619.1 Detailed Description . . . 3640
7.620 algo.h File Reference . . . 3640
7.620.1 Detailed Description . . . 3649
7.621 algobase.h File Reference . . . 3650
7.621.1 Detailed Description . . . 3651
7.622 balanced_quicksort.h File Reference . . . 3652
7.622.1 Detailed Description . . . 3652
7.623 base.h File Reference . . . 3652
7.623.1 Detailed Description . . . 3653
7.624 basic_iterator.h File Reference . . . 3653
7.624.1 Detailed Description . . . 3653
7.625 checkers.h File Reference . . . 3653
7.625.1 Detailed Description . . . 3653
7.626 compiletime_settings.h File Reference . . . 3653
7.626.1 Detailed Description . . . 3654
7.626.2 Macro Definition Documentation . . . 3654

7.627 equally_split.h File Reference . . . 3655
7.627.1 Detailed Description . . . 3655
7.628 features.h File Reference . . . 3655
7.628.1 Detailed Description . . . 3655
7.628.2 Macro Definition Documentation . . . 3655
7.629 find.h File Reference . . . 3657
7.629.1 Detailed Description . . . 3657
7.630 find_selectors.h File Reference . . . 3657
7.630.1 Detailed Description . . . 3658
7.631 for_each.h File Reference . . . 3658
7.631.1 Detailed Description . . . 3658
7.632 for_each_selectors.h File Reference . . . 3658
7.632.1 Detailed Description . . . 3658
7.633 iterator.h File Reference . . . 3659
7.633.1 Detailed Description . . . 3659
7.634 list_partition.h File Reference . . . 3659
7.634.1 Detailed Description . . . 3659
7.635 losertree.h File Reference . . . 3659
7.635.1 Detailed Description . . . 3660
7.636 merge.h File Reference . . . 3660
7.636.1 Detailed Description . . . 3660
7.637 multiseq_selection.h File Reference . . . 3660
7.637.1 Detailed Description . . . 3661
7.638 multiway_merge.h File Reference . . . 3661
7.638.1 Detailed Description . . . 3664
7.638.2 Macro Definition Documentation . . . 3664
7.639 multiway_mergesort.h File Reference . . . 3664
7.639.1 Detailed Description . . . 3664
7.640 numericfwd.h File Reference . . . 3664
7.640.1 Detailed Description . . . 3666
7.641 omp_loop.h File Reference . . . 3666
7.641.1 Detailed Description . . . 3666
7.642 omp_loop_static.h File Reference . . . 3666
7.642.1 Detailed Description . . . 3667
7.643 par_loop.h File Reference . . . 3667
7.643.1 Detailed Description . . . 3667
7.644 parallel.h File Reference . . . 3667
7.644.1 Detailed Description . . . 3667
7.645 partial_sum.h File Reference . . . 3667

7.645.1 Detailed Description . . . 3667
7.646 partition.h File Reference . . . 3668
7.646.1 Detailed Description . . . 3668
7.646.2 Macro Definition Documentation . . . 3668
7.647 queue.h File Reference . . . 3668
7.647.1 Detailed Description . . . 3668
7.647.2 Macro Definition Documentation . . . 3669
7.648 quicksort.h File Reference . . . 3669
7.648.1 Detailed Description . . . 3669
7.649 random_number.h File Reference . . . 3669
7.649.1 Detailed Description . . . 3669
7.650 random_shuffle.h File Reference . . . 3669
7.650.1 Detailed Description . . . 3670
7.651 search.h File Reference . . . 3670
7.651.1 Detailed Description . . . 3670
7.652 set_operations.h File Reference . . . 3670
7.652.1 Detailed Description . . . 3671
7.653 settings.h File Reference . . . 3671
7.653.1 Detailed Description . . . 3671
7.653.2 Deciding whether to run an algorithm in parallel. . . . 3671
7.653.3 Macro Definition Documentation . . . 3672
7.654 sort.h File Reference . . . 3672
7.654.1 Detailed Description . . . 3673
7.655 tags.h File Reference . . . 3673
7.655.1 Detailed Description . . . 3673
7.656 types.h File Reference . . . 3673
7.656.1 Detailed Description . . . 3674
7.657 unique_copy.h File Reference . . . 3674
7.657.1 Detailed Description . . . 3674
7.658 workstealing.h File Reference . . . 3674
7.658.1 Detailed Description . . . 3675
7.659 print File Reference . . . 3675
7.659.1 Detailed Description . . . 3675
7.660 queue File Reference . . . 3675
7.660.1 Detailed Description . . . 3675
7.661 experimental/random File Reference . . . 3675
7.661.1 Detailed Description . . . 3675
7.662 random File Reference . . . 3675
7.662.1 Detailed Description . . . 3675

7.663 ranges File Reference . . . 3676
7.663.1 Detailed Description . . . 3679
7.664 experimental/ratio File Reference . . . 3679
7.664.1 Detailed Description . . . 3679
7.665 ratio File Reference . . . 3679
7.665.1 Detailed Description . . . 3680
7.666 tr2/ratio File Reference . . . 3681
7.666.1 Detailed Description . . . 3681
7.667 experimental/regex File Reference . . . 3681
7.667.1 Detailed Description . . . 3681
7.668 regex File Reference . . . 3681
7.668.1 Detailed Description . . . 3681
7.669 scoped_allocator File Reference . . . 3682
7.669.1 Detailed Description . . . 3682
7.670 semaphore File Reference . . . 3682
7.670.1 Detailed Description . . . 3682
7.671 debug/set File Reference . . . 3682
7.671.1 Detailed Description . . . 3682
7.672 experimental/set File Reference . . . 3682
7.672.1 Detailed Description . . . 3683
7.673 set File Reference . . . 3683
7.673.1 Detailed Description . . . 3683
7.674 shared_mutex File Reference . . . 3683
7.674.1 Detailed Description . . . 3683
7.675 source_location File Reference . . . 3683
7.675.1 Detailed Description . . . 3683
7.676 span File Reference . . . 3683
7.676.1 Detailed Description . . . 3683
7.677 sstream File Reference . . . 3684
7.677.1 Detailed Description . . . 3684
7.678 stack File Reference . . . 3684
7.678.1 Detailed Description . . . 3684
7.679 stdatomic.h File Reference . . . 3684
7.679.1 Detailed Description . . . 3684
7.680 stdbit.h File Reference . . . 3684
7.680.1 Detailed Description . . . 3684
7.681 stdckdint.h File Reference . . . 3685
7.681.1 Detailed Description . . . 3685
7.682 stdexcept File Reference . . . 3685

7.682.1 Detailed Description . . . 3685
7.683 stdlib.h File Reference . . . 3685
7.683.1 Detailed Description . . . 3685
7.683.2 Function Documentation . . . 3685
7.684 stop_token File Reference . . . 3685
7.684.1 Detailed Description . . . 3685
7.685 streambuf File Reference . . . 3685
7.685.1 Detailed Description . . . 3686
7.686 debug/string File Reference . . . 3686
7.686.1 Detailed Description . . . 3688
7.687 experimental/string File Reference . . . 3688
7.687.1 Detailed Description . . . 3689
7.688 string File Reference . . . 3689
7.688.1 Detailed Description . . . 3689
7.689 experimental/string_view File Reference . . . 3689
7.689.1 Detailed Description . . . 3691
7.690 string_view File Reference . . . 3691
7.690.1 Detailed Description . . . 3692
7.691 syncstream File Reference . . . 3692
7.691.1 Detailed Description . . . 3692
7.692 experimental/system_error File Reference . . . 3692
7.692.1 Detailed Description . . . 3693
7.693 system_error File Reference . . . 3693
7.693.1 Detailed Description . . . 3693
7.694 text_encoding File Reference . . . 3693
7.694.1 Detailed Description . . . 3693
7.695 tgmath.h File Reference . . . 3693
7.695.1 Detailed Description . . . 3693
7.696 thread File Reference . . . 3694
7.696.1 Detailed Description . . . 3694
7.697 bool_set File Reference . . . 3694
7.697.1 Detailed Description . . . 3695
7.698 bool_set.tcc File Reference . . . 3695
7.698.1 Detailed Description . . . 3695
7.699 dynamic_bitset File Reference . . . 3695
7.699.1 Detailed Description . . . 3696
7.700 dynamic_bitset.tcc File Reference . . . 3696
7.700.1 Detailed Description . . . 3696
7.701 experimental/tuple File Reference . . . 3696

7.701.1 Detailed Description . . . 3697
7.702 tuple File Reference . . . 3697
7.702.1 Detailed Description . . . 3698
7.703 experimental/type_traits File Reference . . . 3698
7.703.1 Detailed Description . . . 3701
7.704 tr2/type_traits File Reference . . . 3702
7.704.1 Detailed Description . . . 3702
7.705 type_traits File Reference . . . 3702
7.705.1 Detailed Description . . . 3705
7.706 typeindex File Reference . . . 3705
7.706.1 Detailed Description . . . 3705
7.707 debug/unordered_map File Reference . . . 3706
7.707.1 Detailed Description . . . 3708
7.708 experimental/unordered_map File Reference . . . 3708
7.708.1 Detailed Description . . . 3708
7.709 unordered_map File Reference . . . 3708
7.709.1 Detailed Description . . . 3708
7.710 debug/unordered_set File Reference . . . 3709
7.710.1 Detailed Description . . . 3711
7.711 experimental/unordered_set File Reference . . . 3711
7.711.1 Detailed Description . . . 3711
7.712 unordered_set File Reference . . . 3711
7.712.1 Detailed Description . . . 3711
7.713 experimental/utility File Reference . . . 3711
7.713.1 Detailed Description . . . 3712
7.714 utility File Reference . . . 3712
7.714.1 Detailed Description . . . 3712
7.715 valarray File Reference . . . 3712
7.715.1 Detailed Description . . . 3712
7.716 variant File Reference . . . 3712
7.716.1 Detailed Description . . . 3712
7.717 debug/vector File Reference . . . 3712
7.717.1 Detailed Description . . . 3713
7.718 experimental/vector File Reference . . . 3713
7.718.1 Detailed Description . . . 3714
7.719 vector File Reference . . . 3714
7.719.1 Detailed Description . . . 3714
7.720 atomic_word.h File Reference . . . 3714
7.720.1 Detailed Description . . . 3714

7.721 basic_file.h File Reference . . . 3714
7.721.1 Detailed Description . . . 3714
7.722 c++allocator.h File Reference . . . 3714
7.722.1 Detailed Description . . . 3715
7.723 c++config.h File Reference . . . 3715
7.723.1 Detailed Description . . . 3721
7.724 c++io.h File Reference . . . 3721
7.724.1 Detailed Description . . . 3721
7.725 c++locale.h File Reference . . . 3721
7.725.1 Detailed Description . . . 3721
7.726 c++locale_internal.h File Reference . . . 3721
7.726.1 Detailed Description . . . 3722
7.727 parallel/compatibility.h File Reference . . . 3722
7.727.1 Detailed Description . . . 3722
7.728 x86_64-pc-linux-gnu/bits/compatibility.h File Reference . . . 3722
7.728.1 Detailed Description . . . 3722
7.729 cpu_defines.h File Reference . . . 3722
7.729.1 Detailed Description . . . 3722
7.730 ctype_base.h File Reference . . . 3722
7.730.1 Detailed Description . . . 3722
7.731 ctype_inline.h File Reference . . . 3723
7.731.1 Detailed Description . . . 3723
7.732 cxxabi_tweaks.h File Reference . . . 3723
7.732.1 Detailed Description . . . 3723
7.733 error_constants.h File Reference . . . 3723
7.733.1 Detailed Description . . . 3724
7.734 extc++.h File Reference . . . 3724
7.734.1 Detailed Description . . . 3724
7.735 messages_members.h File Reference . . . 3724
7.735.1 Detailed Description . . . 3724
7.736 opt_random.h File Reference . . . 3724
7.736.1 Detailed Description . . . 3724
7.737 os_defines.h File Reference . . . 3724
7.737.1 Detailed Description . . . 3724
7.738 stdc++.h File Reference . . . 3725
7.738.1 Detailed Description . . . 3725
7.739 stdtr1c++.h File Reference . . . 3725
7.739.1 Detailed Description . . . 3725
7.740 time_members.h File Reference . . . 3725

7.740.1 Detailed Description . . . 3725
7.741 compare File Reference . . . 3725
7.741.1 Detailed Description . . . 3726
7.742 cxxabi.h File Reference . . . 3726
7.742.1 Detailed Description . . . 3727
7.742.2 Function Documentation . . . 3727
7.743 exception File Reference . . . 3728
7.743.1 Detailed Description . . . 3729
7.744 initializer_list File Reference . . . 3729
7.744.1 Detailed Description . . . 3729
7.745 new File Reference . . . 3729
7.745.1 Detailed Description . . . 3730
7.745.2 Function Documentation . . . 3730
7.746 typeinfo File Reference . . . 3730
7.746.1 Detailed Description . . . 3731

Index **3733**

1 Deprecated List

Module negators

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

Module pointer_adaptors

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

Module ptrmem_adaptors

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

Struct std::aligned_storage< _Len, _Align >

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

Struct std::aligned_union< _Len, _Types >

Deprecated in C++23.

Class std::auto_ptr< _Tp >

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

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

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

Struct std::is_literal_type< _Tp >

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

Struct std::is_pod< _Tp >

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

Struct std::is_trivial< _Tp >

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

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

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

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

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

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

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

Struct std::unary_function< _Arg, _Result >

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

2 Todo List

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Class __gnu_cxx::rope< _CharT, _Alloc >

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

Class __gnu_cxx::slist< _Tp, _Alloc >

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

Module mathsf

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

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

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

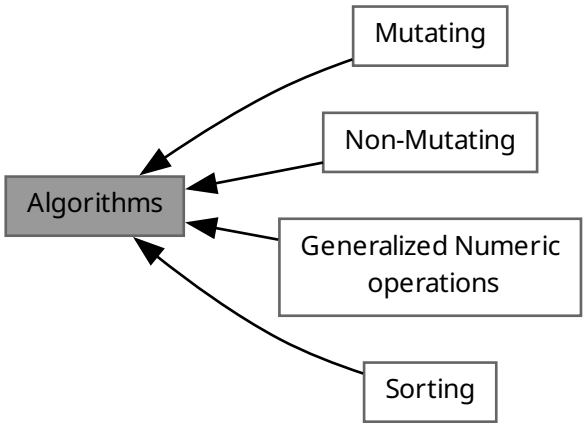
Member std::regex_traits< _Ch_type >::transform_primary (_Fwd_iter __first, _Fwd_iter __last) const

Implement this function correctly.

3 Topic Documentation

3.1 Algorithms

Collaboration diagram for Algorithms:



Topics

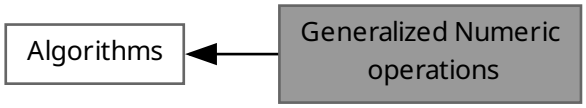
- Generalized Numeric operations
- Mutating
- Non-Mutating
- Sorting

3.1.1 Detailed Description

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

3.1.2 Generalized Numeric operations

Collaboration diagram for Generalized Numeric operations:



Functions

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

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

3.1.2.1 Detailed Description

3.1.2.2 Function Documentation

accumulate() [1/2]

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

Accumulate values in a range.

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

Parameters

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

Returns

The final sum.

accumulate() [2/2]

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

Accumulate values in a range with operation.

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

Parameters

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

Returns

The final sum.

adjacent_difference() [1/2]

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

Return differences between adjacent values.

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

Parameters

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

Returns

Iterator pointing just beyond the values written to result.

adjacent_difference() [2/2]

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

Return differences between adjacent values.

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

Parameters

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

Returns

Iterator pointing just beyond the values written to result.

exclusive_scan() [1/2]

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

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

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

Parameters

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

Returns

The end of the output range.

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

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

References exclusive_scan(), and move().

exclusive_scan() [2/2]

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

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

Parameters

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

Returns

The end of the output range.

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

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

References move().

Referenced by exclusive_scan().

inclusive_scan() [1/3]

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

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

Parameters

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

Returns

The end of the output range.

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

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

References inclusive_scan().

inclusive_scan() [2/3]

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

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

Parameters

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

Returns

The end of the output range.

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

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

References inclusive_scan(), and move().

inclusive_scan() [3/3]

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

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

Parameters

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

__result	Start of output range.
__binary_op	Function to perform summation.
__init	Initial value.

Returns

The end of the output range.

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

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

Referenced by inclusive_scan(), and inclusive_scan().

inner_product() [1/2]

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

Compute inner product of two ranges.

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

Parameters

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

Returns

The final inner product.

inner_product() [2/2]

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

Compute inner product of two ranges.

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

Parameters

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

Returns

The final inner product.

iota()

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

Create a range of sequentially increasing values.

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

Parameters

__first	Start of range.

__last	End of range.
__value	Starting value.

partial_sum() [1/2]

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

Return list of partial sums.

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

Parameters

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

Returns

Iterator pointing just beyond the values written to __result.

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

partial_sum() [2/2]

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

Return list of partial sums.

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

Parameters

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

Returns

Iterator pointing just beyond the values written to __result.

reduce() [1/3]

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

Calculate reduction of values in a range.

Parameters

__first	Start of range.
__last	End of range.

Returns

The final sum.

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

References reduce().

reduce() [2/3]

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

Calculate reduction of values in a range.

Parameters

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

Returns

The final sum.

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

References move(), and reduce().

reduce() [3/3]

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

Calculate reduction of values in a range.

Parameters

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

Returns

The final sum.

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

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

Referenced by reduce(), and reduce().

transform_exclusive_scan()

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

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

Parameters

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

Returns

The end of the output range.

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

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

References move().

transform_inclusive_scan() [1/2]

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

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

Parameters

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

Returns

The end of the output range.

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

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

References move(), and transform_inclusive_scan().

transform_inclusive_scan() [2/2]

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

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

Parameters

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

Returns

The end of the output range.

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

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

Referenced by transform_inclusive_scan().

transform_reduce() [1/3]

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

Transform the elements of a range and reduce.

Parameters

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

Returns

The final sum.

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

transform_reduce() [2/3]

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

Combine elements from two ranges and reduce.

Parameters

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

Returns

The final sum.

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

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

References move(), and transform_reduce().

transform_reduce() [3/3]

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

Combine elements from two ranges and reduce.

Parameters

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

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

Returns

The final sum.

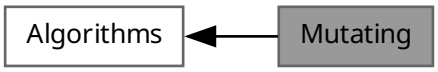
Call `binary_op2(first1[n],first2[n])` for each `n` in `[0,last1-first1)` and then use `binary_op1` to reduce the values returned by `binary_op2` to a single value of type `T`.

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

Referenced by transform_reduce().

3.1.3 Mutating

Collaboration diagram for Mutating:



Functions

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

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

- template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator>
  void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumber↩
  Generator &&__g)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate ↩
  __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _↩
  ForwardIterator2 __first2)
- template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation>
  constexpr _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, ↩
  _UnaryOperation __unary_op)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,
  _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↩
  binary_pred)
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>
  constexpr _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result,
  _BinaryPredicate __binary_pred)

3.1.3.1 Detailed Description

3.1.3.2 Function Documentation

copy()

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

Copies the range [first,last) into result.

Parameters

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

Returns

result + (last - first)

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

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

copy_backward()

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

Copies the range [first,last) into result.

Parameters

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

Returns

result - (last - first)

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

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

copy_if()

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

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

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

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

copy_n()

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

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

Parameters

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

Returns

result+n.

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

References move().

fill()

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

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

Parameters

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

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

fill_n()

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

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

Parameters

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

Returns

The iterator at first+n.

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

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

generate()

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

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

Parameters

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

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

generate_n()

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

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

Parameters

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

Returns

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

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

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

is_partitioned()

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

Checks whether the sequence is partitioned.

Parameters

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

Returns

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

iter_swap()

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

Swaps the contents of two iterators.

Parameters

_a	An iterator.
_b	Another iterator.

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

move()

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

Moves the range [first,last) into result.

Parameters

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

Returns

result + (last - first)

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

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

move_backward()

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

Moves the range [first,last) into result.

Parameters

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

Returns

result - (last - first)

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

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

partition()

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

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

Parameters

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

Returns

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

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

partition_copy()

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

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

Parameters

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

Returns

A pair designating the ends of the resulting sequences.

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

partition_point()

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

Find the partition point of a partitioned range.

Parameters

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

Returns

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

References advance(), and distance().

random_shuffle() [1/2]

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

Randomly shuffle the elements of a sequence.

Parameters

__first	A forward iterator.
__last	A forward iterator.

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

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

random_shuffle() [2/2]

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

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

Parameters

__first	A forward iterator.
__last	A forward iterator.
__rand	The RNG function object.

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

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

remove()

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

Remove elements from a sequence.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

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

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

remove_copy()

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

Copy a sequence, removing elements of a given value.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

remove_copy_if()

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

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

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

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

remove_if()

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

Remove elements from a sequence using a predicate.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

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

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

replace()

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

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

Parameters

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

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

replace_copy_if()

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

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

Parameters

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

Returns

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

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

replace_if()

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

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

Parameters

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

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

reverse()

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

Reverse a sequence.

Parameters

__first	A bidirectional iterator.
__last	A bidirectional iterator.

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

reverse_copy()

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

Copy a sequence, reversing its elements.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

rotate()

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

Rotate the elements of a sequence.

Parameters

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

Returns

first + (last - middle).

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

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

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

rotate_copy()

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

Copy a sequence, rotating its elements.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

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

shuffle()

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

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

Parameters

__first	A forward iterator.
__last	A forward iterator.
__g	A UniformRandomNumberGenerator (C++11 26.5.1.3).

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

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

stable_partition()

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

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

Parameters

__first	A forward iterator.
__last	A forward iterator.

__pred	A predicate function object.

Returns

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

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

swap_ranges()

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

Swap the elements of two sequences.

Parameters

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

Returns

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

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

transform() [1/2]

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

Perform an operation on a sequence.

Parameters

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

Returns

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

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

`__unary_op` must not alter its argument.

transform() [2/2]

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

Perform an operation on corresponding elements of two sequences.

Parameters

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

Returns

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

Applies the operator to the corresponding elements in the two input ranges and assigns the results to successive elements of the output sequence. Evaluates `*(__result+N) = __binary_op(*(__first1+N), *(__↩ first2+N))` for each `N` in the range `[0, __last1-__first1)`.

`__binary_op` must not alter either of its arguments.

unique() [1/2]

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

Remove consecutive duplicate values from a sequence.

Parameters

__first	A forward iterator.
__last	A forward iterator.

Returns

An iterator designating the end of the resulting sequence.

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

unique() [2/2]

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

Remove consecutive values from a sequence using a predicate.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

unique_copy() [1/2]

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

Copy a sequence, removing consecutive duplicate values.

Parameters

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

Returns

An iterator designating the end of the resulting sequence.

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

unique_copy() [2/2]

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

Copy a sequence, removing consecutive values using a predicate.

Parameters

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

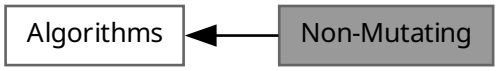
Returns

An iterator designating the end of the resulting sequence.

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

3.1.4 Non-Mutating

Collaboration diagram for Non-Mutating:



Functions

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

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

3.1.4.1 Detailed Description

3.1.4.2 Function Documentation

adjacent_find() [1/2]

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

Find two adjacent values in a sequence that are equal.

Parameters

__first	A forward iterator.
__last	A forward iterator.

Returns

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

adjacent_find() [2/2]

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

Find two adjacent values in a sequence using a predicate.

Parameters

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

Returns

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

all_of()

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

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

Parameters

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

Returns

True if the check is true, false otherwise.

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

any_of()

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

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

Parameters

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

Returns

True if the check is true, false otherwise.

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

count()

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

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

Parameters

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

Returns

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

count_if()

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

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

Parameters

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

Returns

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

equal() [1/4]

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

Tests a range for element-wise equality.

Parameters

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

Returns

A boolean true or false.

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

equal() [2/4]

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

Tests a range for element-wise equality.

Parameters

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

Returns

A boolean true or false.

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

equal() [3/4]

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

Tests a range for element-wise equality.

Parameters

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

Returns

A boolean true or false.

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

equal() [4/4]

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

Tests a range for element-wise equality.

Parameters

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

Returns

A boolean true or false.

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

find()

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

Find the first occurrence of a value in a sequence.

Parameters

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

Returns

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

References to_address().

find_end() [1/2]

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

Find last matching subsequence in a sequence.

Parameters

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

Returns

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

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

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

find_end() [2/2]

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

Find last matching subsequence in a sequence using a predicate.

Parameters

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

Returns

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

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

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

find_first_of() [1/2]

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

Find element from a set in a sequence.

Parameters

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

Returns

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

Searches the range `[__first1, __last1)` for an element that is equal to some element in the range `[_↩ _first2, __last2)`. If found, returns an iterator in the range `[__first1, __last1)`, otherwise returns `__last1`.

find_first_of() [2/2]

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

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

Parameters

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

__comp	Predicate to use.

Returns

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

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

find_if()

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

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

Parameters

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

Returns

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

find_if_not()

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

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

Parameters

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

Returns

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

for_each()

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

Apply a function to every element of a sequence.

Parameters

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

Returns

`__f`

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

for_each_n()

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

Apply a function to every element of a sequence.

Parameters

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

Returns

`__first+__n`

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

References move().

is_permutation() [1/4]

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

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

Parameters

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

Returns

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

is_permutation() [2/4]

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

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

Parameters

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

Returns

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

is_permutation() [3/4]

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

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

Parameters

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

Returns

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

is_permutation() [4/4]

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

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

Parameters

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

Returns

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

mismatch() [1/4]

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

Finds the places in ranges which don't match.

Parameters

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

Returns

A pair of iterators pointing to the first mismatch.

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

mismatch() [2/4]

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

Finds the places in ranges which don't match.

Parameters

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

Returns

A pair of iterators pointing to the first mismatch.

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

mismatch() [3/4]

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

Finds the places in ranges which don't match.

Parameters

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

Returns

A pair of iterators pointing to the first mismatch.

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

mismatch() [4/4]

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

Finds the places in ranges which don't match.

Parameters

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

Returns

A pair of iterators pointing to the first mismatch.

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

none_of()

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

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

Parameters

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

Returns

True if the check is true, false otherwise.

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

search() [1/2]

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

Search a sequence for a matching sub-sequence.

Parameters

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

Returns

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

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

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

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

search() [2/2]

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

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

Parameters

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

Returns

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

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

See also

search(_ForwardIter1, _ForwardIter1, _ForwardIter2, _ForwardIter2)

search_n() [1/2]

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

Search a sequence for a number of consecutive values.

Parameters

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

Returns

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

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

search_n() [2/2]

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

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

Parameters

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

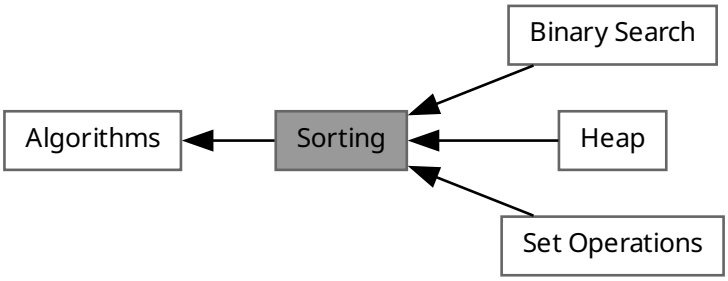
Returns

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

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

3.1.5 Sorting

Collaboration diagram for Sorting:



Topics

- Binary Search
- Heap
- Set Operations

Functions

- template<typename _Tp>
  constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)
- template<typename _BidirectionalIterator>
  constexpr void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _Bidirectional↩ Iterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _Bidirectional↩ Iterator __last, _Compare __comp)
- template<typename _ForwardIterator>
  constexpr bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare _↩ _comp)
- template<typename _II1, typename _II2>
  constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1, typename _II2, typename _Compare>
  constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)
- template<typename _InputIter1, typename _InputIter2, typename _Comp>
  constexpr auto std::lexicographical_compare_three_way (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 _↩ _first2, _InputIter2 __last2, _Comp __comp) -> decltype(__comp(*__first1, *__first2))
- template<typename _Tp>
  constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __↩ comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _Tp>
  constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b)

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

3.1.5.1 Detailed Description

3.1.5.2 Function Documentation

clamp() [1/2]

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

Returns the value clamped between lo and hi.

Parameters

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

Return values

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

Precondition

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

References max(), and min().

clamp() [2/2]

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

Returns the value clamped between lo and hi.

Parameters

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

Return values

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

Precondition

`__comp(__hi, __lo)` is false.

References max(), and min().

inplace_merge() [1/2]

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

Merges two sorted ranges in place.

Parameters

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

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

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

inplace_merge() [2/2]

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

Merges two sorted ranges in place.

Parameters

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

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

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

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

is_sorted() [1/2]

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

Determines whether the elements of a sequence are sorted.

Parameters

__first	An iterator.
__last	Another iterator.

Returns

True if the elements are sorted, false otherwise.

is_sorted() [2/2]

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

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

Parameters

__first	An iterator.
__last	Another iterator.
__comp	A comparison function object.

Returns

True if the elements are sorted, false otherwise.

is_sorted_until() [1/2]

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

Determines the end of a sorted sequence.

Parameters

__first	An iterator.
__last	Another iterator.

Returns

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

is_sorted_until() [2/2]

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

Determines the end of a sorted sequence using comparison function.

Parameters

__first	An iterator.
__last	Another iterator.
__comp	A comparison function object.

Returns

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

lexicographical_compare() [1/2]

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

Performs **dictionary** comparison on ranges.

Parameters

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

Returns

A boolean true or false.

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

lexicographical_compare() [2/2]

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

Performs **dictionary** comparison on ranges.

Parameters

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

Returns

A boolean true or false.

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

lexicographical_compare_three_way()

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

Performs dictionary comparison on ranges.

Parameters

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

Returns

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

Referenced by operator<=>().

max() [1/2]

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

This does what you think it does.

Parameters

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

Returns

The greater of the parameters.

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

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

max() [2/2]

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

This does what you think it does.

Parameters

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

Returns

The greater of the parameters.

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

max_element() [1/2]

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

Return the maximum element in a range.

Parameters

__first	Start of range.
__last	End of range.

Returns

Iterator referencing the first instance of the largest value.

max_element() [2/2]

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

Return the maximum element in a range using comparison function.

Parameters

__first	Start of range.
__last	End of range.
__comp	Comparison function object.

Returns

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

merge() [1/2]

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

Merges two sorted ranges.

Parameters

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

Returns

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

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

merge() [2/2]

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

Merges two sorted ranges.

Parameters

__first1	An iterator.
__first2	Another iterator.

__last1	Another iterator.
__last2	Another iterator.
__result	An iterator pointing to the end of the merged range.
__comp	A function object to use for comparisons.

Returns

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

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

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

min() [1/2]

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

This does what you think it does.

Parameters

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

Returns

The lesser of the parameters.

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

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

min() [2/2]

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

This does what you think it does.

Parameters

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

Returns

The lesser of the parameters.

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

min_element() [1/2]

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

Return the minimum element in a range.

Parameters

__first	Start of range.
__last	End of range.

Returns

Iterator referencing the first instance of the smallest value.

min_element() [2/2]

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

Return the minimum element in a range using comparison function.

Parameters

__first	Start of range.
__last	End of range.
__comp	Comparison function object.

Returns

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

minmax() [1/2]

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

Determines min and max at once as an ordered pair.

Parameters

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

Returns

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

minmax() [2/2]

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

Determines min and max at once as an ordered pair.

Parameters

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

Returns

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

minmax_element() [1/2]

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

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

Parameters

__first	Start of range.
__last	End of range.

Returns

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

minmax_element() [2/2]

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

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

Parameters

__first	Start of range.
__last	End of range.
__comp	Comparison function object.

Returns

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

next_permutation() [1/2]

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

Permute range into the next dictionary ordering.

Parameters

__first	Start of range.
__last	End of range.

Returns

False if wrapped to first permutation, true otherwise.

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

next_permutation() [2/2]

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

Permute range into the next dictionary ordering using a comparison function.

Parameters

__first	Start of range.
__last	End of range.
__comp	A comparison function object.

Returns

False if wrapped to first permutation, true otherwise.

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

nth_element() [1/2]

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

Sort a sequence just enough to find a particular position.

Parameters

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

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

References __lg().

nth_element() [2/2]

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

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

Parameters

__first	An iterator.
__nth	Another iterator.
__last	Another iterator.
__comp	A comparison function object.

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

References __lg().

partial_sort() [1/2]

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

Sort the smallest elements of a sequence.

Parameters

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

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

partial_sort() [2/2]

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

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

Parameters

__first	An iterator.
__middle	Another iterator.
__last	Another iterator.
__comp	A comparison function object.

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

partial_sort_copy() [1/2]

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

Copy the smallest elements of a sequence.

Parameters

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

Returns

An iterator indicating the end of the resulting sequence.

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

partial_sort_copy() [2/2]

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

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

Parameters

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

Returns

An iterator indicating the end of the resulting sequence.

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

prev_permutation() [1/2]

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

Permute range into the previous *dictionary* ordering.

Parameters

__first	Start of range.
__last	End of range.

Returns

False if wrapped to last permutation, true otherwise.

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

prev_permutation() [2/2]

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

Permute range into the previous *dictionary* ordering using a comparison function.

Parameters

__first	Start of range.
__last	End of range.
__comp	A comparison function object.

Returns

False if wrapped to last permutation, true otherwise.

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

sort() [1/2]

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

Sort the elements of a sequence.

Parameters

__first	An iterator.
__last	Another iterator.

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

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

sort() [2/2]

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

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

Parameters

__first	An iterator.
__last	Another iterator.
__comp	A comparison function object.

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

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

stable_sort()

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

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

Parameters

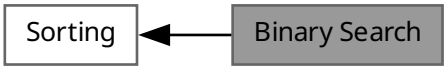
__first	An iterator.
__last	Another iterator.

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

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

3.1.5.3 Binary Search

Collaboration diagram for Binary Search:



Functions

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

3.1.5.3.1 Detailed Description

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

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

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

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

3.1.5.3.2 Function Documentation

binary_search() [1/2]

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

Determines whether an element exists in a range.

Parameters

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

Returns

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

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

binary_search() [2/2]

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

Determines whether an element exists in a range.

Parameters

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

Returns

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

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

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

equal_range() [1/2]

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

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

Parameters

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

Returns

An pair of iterators defining the subrange.

This is equivalent to

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

but does not actually call those functions.

equal_range() [2/2]

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

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

Parameters

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

Returns

An pair of iterators defining the subrange.

This is equivalent to

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

but does not actually call those functions.

lower_bound() [1/2]

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

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

Parameters

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

Returns

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

lower_bound() [2/2]

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

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

Parameters

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

Returns

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

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

upper_bound() [1/2]

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

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

Parameters

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

Returns

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

upper_bound() [2/2]

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

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

Parameters

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

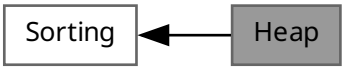
Returns

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

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

3.1.5.4 Heap

Collaboration diagram for Heap:



Functions

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

3.1.5.4.1 Detailed Description

3.1.5.4.2 Function Documentation

is_heap() [1/2]

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

Determines whether a range is a heap.

Parameters

__first	Start of range.
__last	End of range.

Returns

True if range is a heap, false otherwise.

is_heap() [2/2]

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

Determines whether a range is a heap using comparison functor.

Parameters

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

Returns

True if range is a heap, false otherwise.

References distance().

is_heap_until() [1/2]

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

Search the end of a heap.

Parameters

__first	Start of range.
__last	End of range.

Returns

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

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

References distance().

is_heap_until() [2/2]

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

Search the end of a heap using comparison functor.

Parameters

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

Returns

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

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

References distance().

make_heap() [1/2]

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

Construct a heap over a range.

Parameters

__first	Start of heap.
__last	End of heap.

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

make_heap() [2/2]

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

Construct a heap over a range using comparison functor.

Parameters

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

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

pop_heap() [1/2]

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

Pop an element off a heap.

Parameters

__first	Start of heap.
__last	End of heap.

Precondition

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

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

pop_heap() [2/2]

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

Pop an element off a heap using comparison functor.

Parameters

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

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

push_heap() [1/2]

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

Push an element onto a heap.

Parameters

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

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

push_heap() [2/2]

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

Push an element onto a heap using comparison functor.

Parameters

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

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

sort_heap() [1/2]

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

Sort a heap.

Parameters

__first	Start of heap.
__last	End of heap.

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

sort_heap() [2/2]

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

Sort a heap using comparison functor.

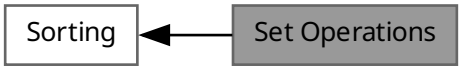
Parameters

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

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

3.1.5.5 Set Operations

Collaboration diagram for Set Operations:



Functions

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

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

3.1.5.5.1 Detailed Description

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

3.1.5.5.2 Function Documentation

includes() [1/2]

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

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

Parameters

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

Returns

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

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

includes() [2/2]

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

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

Parameters

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

Returns

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

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

set_difference() [1/2]

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

Return the difference of two sorted ranges.

Parameters

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

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

Returns

End of the output range.

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

set_difference() [2/2]

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

Return the difference of two sorted ranges using comparison function.

Parameters

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

Returns

End of the output range.

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

set_intersection() [1/2]

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

Return the intersection of two sorted ranges.

Parameters

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

Returns

End of the output range.

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

set_intersection() [2/2]

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

Return the intersection of two sorted ranges using comparison function.

Parameters

__first1	Start of first range.

__last1	End of first range.
__first2	Start of second range.
__last2	End of second range.
__result	Start of output range.
__comp	The comparison function object.

Returns

End of the output range.

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

set_symmetric_difference() [1/2]

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

Return the symmetric difference of two sorted ranges.

Parameters

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

Returns

End of the output range.

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

set_symmetric_difference() [2/2]

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

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

Parameters

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

Returns

End of the output range.

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

set_union() [1/2]

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

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

Parameters

__first	An iterator.
__last	Another iterator.
__comp	A comparison function object.

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

The relative ordering of equivalent elements is preserved, so any two elements `x` and `y` in the range `[__first, __last)` such that `__comp(x, y)` is false and `__comp(y, x)` is false will have the same relative ordering after calling `stable_sort.` */ template<typename _RandomAccessIterator, typename _Compare> constexpr inline void stable_sort(_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp) { concept requirements (_Mutable_RandomAccessIteratorConcept< _RandomAccessIterator>) (_BinaryPredicateConcept<_Compare, typename iterator_traits<_RandomAccessIterator>::value_type, typename iterator_traits<_RandomAccessIterator>::value_type>) ; ;

__stable_sort(__first, __last, __comp); }

template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare> constexpr _OutputIterator __set_union(_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp) { while (__first1 != __last1 && __first2 != __last2) { if (__comp(*__first1, *__first2)) { __result = *__first1; ++__first1; } else if (__comp(*__first2, *__first1)) { __result = *__first2; ++__first2; } else { __result = *__first1; ++__first1; ++__first2; } ++__result; } return std::copy(__first2, __last2, std::copy(__first1, __last1, __result)); }

/**

Return the union of two sorted ranges.

Parameters

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

Returns

End of the output range.

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

set_union() [2/2]

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

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

Parameters

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

Returns

End of the output range.

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

3.2 Atomics

Classes

- struct std::atomic< _Tp >
- struct std::atomic< _Tp * >
- struct std::atomic< bool >
- struct std::atomic< char >
- struct std::atomic< char16_t >
- struct std::atomic< char32_t >
- struct std::atomic< int >
- struct std::atomic< long >
- struct std::atomic< long long >
- struct std::atomic< short >

- struct std::atomic< signed char >
- struct std::atomic< unsigned char >
- struct std::atomic< unsigned int >
- struct std::atomic< unsigned long >
- struct std::atomic< unsigned long long >
- struct std::atomic< unsigned short >
- struct std::atomic< wchar_t >
- struct std::atomic_flag

Macros

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

Typedefs

- typedef unsigned char **std::__atomic_flag_data_type**
- typedef atomic< bool > std::atomic_bool
- typedef atomic< char > std::atomic_char
- typedef atomic< char16_t > std::atomic_char16_t
- typedef atomic< char32_t > std::atomic_char32_t
- typedef atomic< int > std::atomic_int
- typedef atomic< int16_t > std::atomic_int16_t
- typedef atomic< int32_t > std::atomic_int32_t
- typedef atomic< int64_t > std::atomic_int64_t
- typedef atomic< int8_t > std::atomic_int8_t
- typedef atomic< int_fast16_t > std::atomic_int_fast16_t
- typedef atomic< int_fast32_t > std::atomic_int_fast32_t
- typedef atomic< int_fast64_t > std::atomic_int_fast64_t
- typedef atomic< int_fast8_t > std::atomic_int_fast8_t
- typedef atomic< int_least16_t > std::atomic_int_least16_t
- typedef atomic< int_least32_t > std::atomic_int_least32_t
- typedef atomic< int_least64_t > std::atomic_int_least64_t
- typedef atomic< int_least8_t > std::atomic_int_least8_t
- typedef atomic< intmax_t > std::atomic_intmax_t
- typedef atomic< intptr_t > std::atomic_intptr_t
- typedef atomic< long long > std::atomic_llong
- typedef atomic< long > std::atomic_long
- typedef atomic< ptrdiff_t > std::atomic_ptrdiff_t

- typedef atomic< signed char > std::atomic_schar
- typedef atomic< short > std::atomic_short
- typedef atomic< size_t > std::atomic_size_t
- typedef atomic< unsigned char > std::atomic_uchar
- typedef atomic< unsigned int > std::atomic_uint
- typedef atomic< uint16_t > std::atomic_uint16_t
- typedef atomic< uint32_t > std::atomic_uint32_t
- typedef atomic< uint64_t > std::atomic_uint64_t
- typedef atomic< uint8_t > std::atomic_uint8_t
- typedef atomic< uint_fast16_t > std::atomic_uint_fast16_t
- typedef atomic< uint_fast32_t > std::atomic_uint_fast32_t
- typedef atomic< uint_fast64_t > std::atomic_uint_fast64_t
- typedef atomic< uint_fast8_t > std::atomic_uint_fast8_t
- typedef atomic< uint_least16_t > std::atomic_uint_least16_t
- typedef atomic< uint_least32_t > std::atomic_uint_least32_t
- typedef atomic< uint_least64_t > std::atomic_uint_least64_t
- typedef atomic< uint_least8_t > std::atomic_uint_least8_t
- typedef atomic< uintmax_t > std::atomic_uintmax_t
- typedef atomic< uintptr_t > std::atomic_uintptr_t
- typedef atomic< unsigned long long > std::atomic_ullong
- typedef atomic< unsigned long > std::atomic_ulong
- typedef atomic< unsigned short > std::atomic_ushort
- typedef atomic< wchar_t > std::atomic_wchar_t

Enumerations

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

Functions

- template<typename _ITp>
  bool **std::atomic_compare_exchange_strong** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __↩
  atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_strong** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1,
  __atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_strong_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1,
  __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_strong_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp
  > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_weak** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __↩
  atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_weak** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1,
  __atomic_val_t< _ITp > __i2) noexcept

- template<typename _ITp>
  bool **std::atomic_compare_exchange_weak_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  bool **std::atomic_compare_exchange_weak_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  _ITp **std::atomic_exchange** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_exchange** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_exchange_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_exchange_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_add** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_add** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_add_explicit** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_add_explicit** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_and** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_and** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_and_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_and_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_or** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_or** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_or_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_or_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_sub** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_sub** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_sub_explicit** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept

- template<typename _ITp>
  _ITp **std::atomic_fetch_sub_explicit** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_xor** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_xor** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_xor_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_fetch_xor_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- void **std::atomic_flag_clear** (atomic_flag *__a) noexcept
- void **std::atomic_flag_clear** (volatile atomic_flag *__a) noexcept
- void **std::atomic_flag_clear_explicit** (atomic_flag *__a, memory_order __m) noexcept
- void **std::atomic_flag_clear_explicit** (volatile atomic_flag *__a, memory_order __m) noexcept
- bool **std::atomic_flag_test_and_set** (atomic_flag *__a) noexcept
- bool **std::atomic_flag_test_and_set** (volatile atomic_flag *__a) noexcept
- bool **std::atomic_flag_test_and_set_explicit** (atomic_flag *__a, memory_order __m) noexcept
- bool **std::atomic_flag_test_and_set_explicit** (volatile atomic_flag *__a, memory_order __m) noexcept
- template<typename _ITp>
  void **std::atomic_init** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **std::atomic_init** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  bool **std::atomic_is_lock_free** (const atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  bool **std::atomic_is_lock_free** (const volatile atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **std::atomic_load** (const atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **std::atomic_load** (const volatile atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **std::atomic_load_explicit** (const atomic< _ITp > *__a, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_load_explicit** (const volatile atomic< _ITp > *__a, memory_order __m) noexcept
- void **std::atomic_signal_fence** (memory_order __m) noexcept
- template<typename _ITp>
  void **std::atomic_store** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **std::atomic_store** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **std::atomic_store_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  void **std::atomic_store_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- void **std::atomic_thread_fence** (memory_order __m) noexcept
- template<typename _Tp>
  _Tp std::kill_dependency (_Tp __y) noexcept
- constexpr memory_order **std::operator&** (memory_order __m, __memory_order_modifier __mod) noexcept
- constexpr memory_order **std::operator|** (memory_order __m, __memory_order_modifier __mod) noexcept

Variables

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

3.2.1 Detailed Description

Components for performing atomic operations.

3.2.2 Macro Definition Documentation

ATOMIC_BOOL_LOCK_FREE

```
#define ATOMIC_BOOL_LOCK_FREE
```

Lock-free property.

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

3.2.3 Typedef Documentation

atomic_bool

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

atomic_bool

atomic_char

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

atomic_char

atomic_char16_t

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

atomic_char16_t

atomic_char32_t

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

atomic_char32_t

atomic_int

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

atomic_int

atomic_int16_t

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

atomic_int16_t

atomic_int32_t

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

atomic_int32_t

atomic_int64_t

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

atomic_int64_t

atomic_int8_t

```
typedef atomic<int8_t> std::atomic_int8_t
```

atomic_int8_t

atomic_int_fast16_t

```
typedef atomic<int_fast16_t> std::atomic_int_fast16_t
```

atomic_int_fast16_t

atomic_int_fast32_t

```
typedef atomic<int_fast32_t> std::atomic_int_fast32_t
```

atomic_int_fast32_t

atomic_int_fast64_t

```
typedef atomic<int_fast64_t> std::atomic_int_fast64_t
```

atomic_int_fast64_t

atomic_int_fast8_t

```
typedef atomic<int_fast8_t> std::atomic_int_fast8_t
```

atomic_int_fast8_t

atomic_int_least16_t

```
typedef atomic<int_least16_t> std::atomic_int_least16_t
```

atomic_int_least16_t

atomic_int_least32_t

```
typedef atomic<int_least32_t> std::atomic_int_least32_t
```

atomic_int_least32_t

atomic_int_least64_t

```
typedef atomic<int_least64_t> std::atomic_int_least64_t
```

atomic_int_least64_t

atomic_int_least8_t

```
typedef atomic<int_least8_t> std::atomic_int_least8_t
```

atomic_int_least8_t

atomic_intmax_t

```
typedef atomic<intmax_t> std::atomic_intmax_t
```

atomic_intmax_t

atomic_intptr_t

```
typedef atomic<intptr_t> std::atomic_intptr_t
```

atomic_intptr_t

atomic_llong

```
typedef atomic<long long> std::atomic_llong
```

atomic_llong

atomic_long

```
typedef atomic<long> std::atomic_long
```

atomic_long

atomic_ptrdiff_t

```
typedef atomic<ptrdiff_t> std::atomic_ptrdiff_t
```

atomic_ptrdiff_t

atomic_schar

```
typedef atomic<signed char> std::atomic_schar
```

atomic_schar

atomic_short

```
typedef atomic<short> std::atomic_short
```

atomic_short

atomic_size_t

```
typedef atomic<size_t> std::atomic_size_t
```

atomic_size_t

atomic_uchar

```
typedef atomic<unsigned char> std::atomic_uchar
```

atomic_uchar

atomic_uint

```
typedef atomic<unsigned int> std::atomic_uint
```

atomic_uint

atomic_uint16_t

```
typedef atomic<uint16_t> std::atomic_uint16_t
```

atomic_uint16_t

atomic_uint32_t

```
typedef atomic<uint32_t> std::atomic_uint32_t
```

atomic_uint32_t

atomic_uint64_t

```
typedef atomic<uint64_t> std::atomic_uint64_t
```

atomic_uint64_t

atomic_uint8_t

```
typedef atomic<uint8_t> std::atomic_uint8_t
```

atomic_uint8_t

atomic_uint_fast16_t

```
typedef atomic<uint_fast16_t> std::atomic_uint_fast16_t
```

atomic_uint_fast16_t

atomic_uint_fast32_t

```
typedef atomic<uint_fast32_t> std::atomic_uint_fast32_t
```

atomic_uint_fast32_t

atomic_uint_fast64_t

```
typedef atomic<uint_fast64_t> std::atomic_uint_fast64_t
```

atomic_uint_fast64_t

atomic_uint_fast8_t

```
typedef atomic<uint_fast8_t> std::atomic_uint_fast8_t
```

atomic_uint_fast8_t

atomic_uint_least16_t

```
typedef atomic<uint_least16_t> std::atomic_uint_least16_t
```

atomic_uint_least16_t

atomic_uint_least32_t

```
typedef atomic<uint_least32_t> std::atomic_uint_least32_t
```

atomic_uint_least32_t

atomic_uint_least64_t

```
typedef atomic<uint_least64_t> std::atomic_uint_least64_t
```

atomic_uint_least64_t

atomic_uint_least8_t

```
typedef atomic<uint_least8_t> std::atomic_uint_least8_t
```

atomic_uint_least8_t

atomic_uintmax_t

```
typedef atomic<uintmax_t> std::atomic_uintmax_t
```

atomic_uintmax_t

atomic_uintptr_t

```
typedef atomic<uintptr_t> std::atomic_uintptr_t
```

atomic_uintptr_t

atomic_ullong

```
typedef atomic<unsigned long long> std::atomic_ullong
```

atomic_ullong

atomic_ulong

```
typedef atomic<unsigned long> std::atomic_ulong
```

atomic_ulong

atomic_ushort

```
typedef atomic<unsigned short> std::atomic_ushort
```

atomic_ushort

atomic_wchar_t

```
typedef atomic<wchar_t> std::atomic_wchar_t
```

atomic_wchar_t

3.2.4 Enumeration Type Documentation

memory_order

```
enum class std::memory_order :  int  [strong]
```

Enumeration for memory_order.

3.2.5 Function Documentation

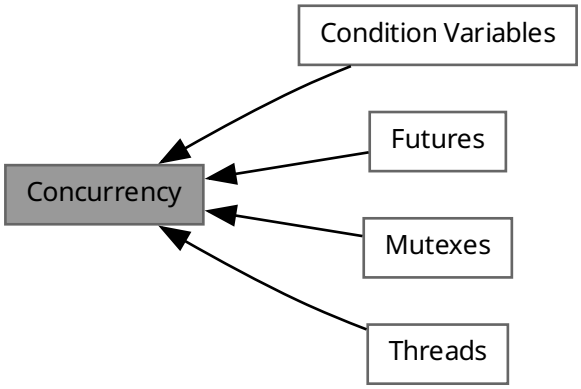
kill_dependency()

```
template<typename _Tp>
_Tp std::kill_dependency (
             _Tp __y)  [inline], [noexcept]
```

kill_dependency

3.3 Concurrency

Collaboration diagram for Concurrency:



Topics

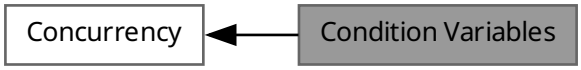
- Condition Variables
- Futures
- Mutexes
- Threads

3.3.1 Detailed Description

Components for concurrent operations, including threads, mutexes, and condition variables.

3.3.2 Condition Variables

Collaboration diagram for Condition Variables:



Classes

- class std::condition_variable
- class std::condition_variable_any

Enumerations

- enum class std::cv_status { **no_timeout** , **timeout** }

Functions

- void **std::notify_all_at_thread_exit** (condition_variable &, unique_lock< mutex >)

3.3.2.1 Detailed Description

Classes for condition_variable support.

3.3.2.2 Enumeration Type Documentation

cv_status

```
enum class std::cv_status  [strong]
```

cv_status

3.3.3 Futures

Collaboration diagram for Futures:



Classes

- class std::__basic_future< _Res >
- class std::future< _Res >
- class std::future< _Res & >
- class std::future< void >
- class std::future_error
- struct std::is_error_code_enum< future_errc >
- class std::packaged_task< _Res(_ArgTypes...)>
- class std::promise< _Res >
- class std::promise< _Res & >
- class std::promise< void >
- class std::shared_future< _Res >
- class std::shared_future< _Res & >
- class std::shared_future< void >

Enumerations

- enum class std::future_errc { **future_already_retrieved** , **promise_already_satisfied** , **no_state** , **broken_promise** }
- enum class std::future_status { **ready** , **timeout** , **deferred** }
- enum class std::launch { **async** , **deferred** }

Functions

- **std::__basic_future< _Res >::__basic_future** (const shared_future< _Res > &) noexcept
- **std::__basic_future< _Res >::__basic_future** (future< _Res > &&) noexcept
- **std::__basic_future< _Res >::__basic_future** (shared_future< _Res > &&) noexcept
- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > std::async (_Fn &&__fn, _Args &&... __args)
- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > std::async (launch __policy, _Fn &&__fn, _Args &&... __args)
- const error_category & std::future_category () noexcept
- error_code std::make_error_code (future_errc __errc) noexcept
- error_condition std::make_error_condition (future_errc __errc) noexcept

- constexpr launch **std::operator&** (launch __x, launch __y) noexcept
- constexpr launch & **std::operator&=** (launch &__x, launch __y) noexcept
- constexpr launch **std::operator^** (launch __x, launch __y) noexcept
- constexpr launch & **std::operator^=** (launch &__x, launch __y) noexcept
- constexpr launch **std::operator|** (launch __x, launch __y) noexcept
- constexpr launch & **std::operator|=** (launch &__x, launch __y) noexcept
- constexpr launch **std::operator~** (launch __x) noexcept
- template<typename _Fun, typename _Signature = __function_guide_t<_Fun, decltype(&_Fun::operator())>>
  std::packaged_task (_Fun) -> packaged_task< _Signature >
- template<typename _Res, typename... _ArgTypes>
  std::packaged_task (_Res(*)(_ArgTypes...)) -> packaged_task< _Res(_ArgTypes...)>
- shared_future< _Res > **std::future< _Res >::share** () noexcept
- shared_future< _Res & > **std::future< _Res & >::share** () noexcept
- shared_future< void > **std::future< void >::share** () noexcept
- template<typename _Res, typename... _ArgTypes>
  void std::swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y) noexcept
- template<typename _Res>
  void **std::swap** (promise< _Res > &__x, promise< _Res > &__y) noexcept

3.3.3.1 Detailed Description

Futures and promises provide support for retrieving the result from an asynchronous function, e.g. one that is running in another thread. A `std::future` represents an asynchronous result that will become ready at some later time. A consumer can wait on a future until the result is ready to be accessed.

Since

C++11

3.3.3.2 Enumeration Type Documentation

future_errc

```
enum class std::future_errc  [strong]
```

Error code for futures.

future_status

```
enum class std::future_status  [strong]
```

Status code for futures.

launch

```
enum class std::launch  [strong]
```

Launch code for futures.

3.3.3.3 Function Documentation

async() [1/2]

```
template<typename _Fn, typename...  _Args>
future< __async_result_of< _Fn, _Args...  > > std::async (
            _Fn && __fn,
            _Args &&...  __args)  [inline], [nodiscard]
```

async, potential overload

References async(), and forward().

async() [2/2]

```
template<typename _Fn, typename...  _Args>
future< __async_result_of< _Fn, _Args...  > > std::async (
            launch __policy,
            _Fn && __fn,
            _Args &&...  __args)  [nodiscard]
```

async

References async(), forward(), and move().

Referenced by async(), and async().

future_category()

```
const error_category & std::future_category ()  [noexcept]
```

Points to a statically-allocated object derived from error_category.

Referenced by make_error_code(), and make_error_condition().

make_error_code()

```
error_code std::make_error_code (
            future_errc __errc)  [inline], [noexcept]
```

Overload of make_error_code for `future_errc`.

References future_category(), and make_error_code().

Referenced by make_error_code(), system_category(), and std::filesystem::filesystem_error::what().

make_error_condition()

```
error_condition std::make_error_condition (
            future_errc __errc)  [inline], [noexcept]
```

Overload of make_error_condition for `future_errc`.

References future_category().

Referenced by system_category().

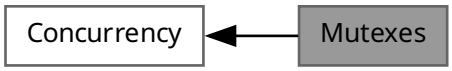
swap()

```
template<typename _Res, typename...  _ArgTypes>
void std::swap (
            packaged_task< _Res(_ArgTypes...)> & __x,
            packaged_task< _Res(_ArgTypes...)> & __y)  [inline], [noexcept]
```

swap

3.3.4 Mutexes

Collaboration diagram for Mutexes:



Classes

- struct std::adopt_lock_t
- struct std::defer_lock_t
- class std::lock_guard< _Mutex >
- class std::mutex
- struct std::once_flag
- class std::recursive_mutex
- class std::recursive_timed_mutex
- class std::shared_lock< _Mutex >
- class std::shared_timed_mutex
- class std::timed_mutex
- struct std::try_to_lock_t
- class std::unique_lock< _Mutex >

Functions

- void **std::__once_proxy** (void)
- template<typename _Callable, typename... _Args>
  void std::call_once (once_flag &__once, _Callable &&__f, _Args &&... __args)
- template<typename _L1, typename _L2, typename... _L3>
  void std::lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)
- template<typename _L1, typename _L2, typename... _L3>
  int std::try_lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)

Variables

- constexpr adopt_lock_t std::adopt_lock
- constexpr defer_lock_t std::defer_lock
- constexpr try_to_lock_t std::try_to_lock

3.3.4.1 Detailed Description

Classes for mutex support.

3.3.4.2 Function Documentation

call_once()

```
template<typename _Callable, typename...  _Args>
void std::call_once (
            once_flag & __once,
            _Callable && __f,
            _Args &&...  __args)
```

Invoke a callable and synchronize with other calls using the same flag.

lock()

```
template<typename _L1, typename _L2, typename...  _L3>
void std::lock (
            _L1 & __l1,
            _L2 & __l2,
            _L3 &...  __l3)
```

Generic lock.

Parameters

_↩ _l1	Meets Lockable requirements (try_lock() may throw).
_↩ _l2	Meets Lockable requirements (try_lock() may throw).
_↩ _l3	Meets Lockable requirements (try_lock() may throw).

Exceptions

An	exception thrown by an argument's lock() or try_lock() member.

Postcondition

All arguments are locked.

All arguments are locked via a sequence of calls to lock(), try_lock() and unlock(). If this function exits via an exception any locks that were obtained will be released.

References defer_lock, and try_lock().

try_lock()

```
template<typename _L1, typename _L2, typename...  _L3>
int std::try_lock (
             _L1 & __l1,
             _L2 & __l2,
             _L3 &...  __l3)  [inline], [nodiscard]
```

Generic try_lock.

Parameters

_↩ _l1	Meets Lockable requirements (try_lock() may throw).
_↩ _l2	Meets Lockable requirements (try_lock() may throw).
_↩ _l3	Meets Lockable requirements (try_lock() may throw).

Returns

Returns -1 if all try_lock() calls return true. Otherwise returns a 0-based index corresponding to the argument that returned false.

Postcondition

Either all arguments are locked, or none will be.

Sequentially calls try_lock() on each argument.

Referenced by lock().

3.3.4.3 Variable Documentation

adopt_lock

```
adopt_lock_t std::adopt_lock  [inline], [constexpr]
```

Tag used to make a scoped lock take ownership of a locked mutex.

defer_lock

```
defer_lock_t std::defer_lock  [inline], [constexpr]
```

Tag used to prevent a scoped lock from acquiring ownership of a mutex.

Referenced by lock().

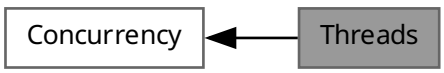
try_to_lock

```
try_to_lock_t std::try_to_lock  [inline], [constexpr]
```

Tag used to prevent a scoped lock from blocking if a mutex is locked.

3.3.5 Threads

Collaboration diagram for Threads:



Namespaces

- namespace std::this_thread

Classes

- struct std::hash< thread::id >
- class std::thread

Functions

- bool **operator==** (thread::id __x, thread::id __y) noexcept
- void **swap** (thread &__x, thread &__y) noexcept

- strong_ordering **operator**<=> (thread::id __x, thread::id __y) noexcept
- template<class _CharT, class _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__out, thread::id __id)

3.3.5.1 Detailed Description

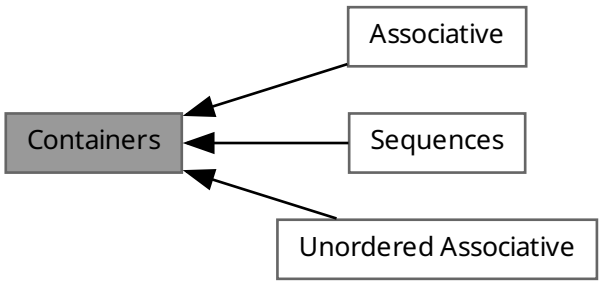
Since

C++11

Classes for thread support.

3.4 Containers

Collaboration diagram for Containers:



Topics

- Associative
- Sequences
- Unordered Associative

3.4.1 Detailed Description

Containers are collections of objects.

A container may hold any type which meets certain requirements, but the type of contained object is chosen at compile time, and all objects in a given container must be of the same type. (Polymorphism is possible by declaring a container of pointers to a base class and then populating it with pointers to instances of derived classes. Variant value types such as the `any` class from `Boost` can also be used.

All contained types must be `Assignable` and `CopyConstructible`. Specific containers may place additional requirements on the types of their contained objects.

Containers manage memory allocation and deallocation themselves when storing your objects. The objects are destroyed when the container is itself destroyed. Note that if you are storing pointers in a container, `delete` is *not* automatically called on the pointers before destroying them.

All containers must meet certain requirements, summarized in `tables`.

The standard containers are further refined into Sequences and Associative Containers. Unordered Associative Containers.

3.4.2 Associative

Collaboration diagram for Associative:



Classes

- class std::map< _Key, _Tp, _Compare, _Alloc >
- class std::multimap< _Key, _Tp, _Compare, _Alloc >
- class std::multiset< _Key, _Compare, _Alloc >
- class std::set< _Key, _Compare, _Alloc >

3.4.2.1 Detailed Description

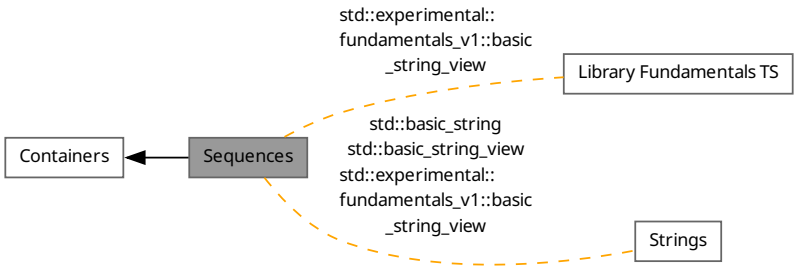
Associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, and an ordering relation used to sort the elements of the container.

All associative containers must meet certain requirements, summarized in `tables`.

3.4.3 Sequences

Collaboration diagram for Sequences:



Files

- file inplace_vector

Classes

- struct std::array< _Tp, _Nm >
- class std::basic_string< _CharT, _Traits, _Alloc >
- class std::basic_string_view< _CharT, _Traits >
- class std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >
- class std::deque< _Tp, _Alloc >
- class std::forward_list< _Tp, _Alloc >
- class std::list< _Tp, _Alloc >
- class std::priority_queue< _Tp, _Sequence, _Compare >
- class std::queue< _Tp, _Sequence >
- class std::stack< _Tp, _Sequence >
- class std::vector< _Tp, _Alloc >
- class std::vector< bool, _Alloc >

3.4.3.1 Detailed Description

Sequences arrange a collection of objects into a strictly linear order.

The differences between sequences are usually due to one or both of the following:

- memory management
- algorithmic complexity

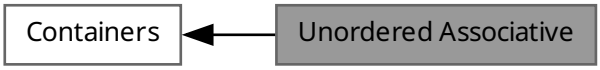
As an example of the first case, `vector` is required to use a contiguous memory layout, while other sequences such as `deque` are not.

The prime reason for choosing one sequence over another should be based on the second category of differences, algorithmic complexity. For example, if you need to perform many inserts and removals from the middle of a sequence, `list` would be ideal. But if you need to perform constant-time access to random elements of the sequence, then `list` should not be used.

All sequences must meet certain requirements, summarized in `tables`.

3.4.4 Unordered Associative

Collaboration diagram for Unordered Associative:



Classes

- class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
- class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >
- class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
- class std::unordered_set< _Value, _Hash, _Pred, _Alloc >

3.4.4.1 Detailed Description

Unordered associative containers allow fast retrieval of data based on keys.

Each container type is parameterized on a `Key` type, a `Hash` type providing a hashing functor, and an ordering relation used to sort the elements of the container.

All unordered associative containers must meet certain requirements, summarized in `tables`.

3.5 Diagnostics

Collaboration diagram for Diagnostics:



Topics

- Exceptions

Classes

- class std::error_category
- class std::error_code
- class std::error_condition
- struct std::is_error_code_enum< _Tp >
- struct std::is_error_condition_enum< _Tp >

Functions

- const error_category & std::generic_category () noexcept
- error_code make_error_code (errc __e) noexcept
- error_condition make_error_condition (errc __e) noexcept
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const error_code &__e)
- strong_ordering operator<=> (const error_code &__lhs, const error_code &__rhs) noexcept
- strong_ordering operator<=> (const error_condition &__lhs, const error_condition &__rhs) noexcept
- bool operator== (const error_code &__lhs, const error_code &__rhs) noexcept
- bool operator== (const error_code &__lhs, const error_condition &__rhs) noexcept
- bool operator== (const error_condition &__lhs, const error_condition &__rhs) noexcept
- const error_category & std::system_category () noexcept

Variables

- template<typename _Tp>
  constexpr bool **std::is_error_code_enum_v**
- template<typename _Tp>
  constexpr bool **std::is_error_condition_enum_v**

3.5.1 Detailed Description

Components for error handling, reporting, and diagnostic operations.

3.5.2 Function Documentation

generic_category()

```
const error_category & std::generic_category ()  [noexcept]
```

Error category for `errno` error codes.

Referenced by std::error_condition::error_condition(), std::error_condition::clear(), std::error_code::make_error_code(), and std::error_condition::make_error_condition().

make_error_code()

```
error_code make_error_code (
            errc __e)  [related]
```

Create an `error_code` representing a standard `errc` condition.

The `std::errc` constants correspond to `errno` macros and so use the generic category.

Since

C++11

References std::generic_category().

Referenced by error_code().

make_error_condition()

```
error_condition make_error_condition (
            errc __e)  [related]
```

Create an `error_condition` representing a standard `errc` condition.

The `std::errc` constants correspond to `errno` macros and so use the generic category.

Since

C++11

References error_condition(), and std::generic_category().

Referenced by error_condition().

operator<<()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const error_code & __e)  [related]
```

Write a std::error_code to an ostream.

Since

C++11

References operator<=>().

operator<=>() [1/2]

```
strong_ordering operator<=> (
            const error_code & __lhs,
            const error_code & __rhs)  [related]
```

Ordered comparison for std::error_code.

This defines a total order by comparing the categories, and then if they are equal comparing the values.

Since

C++11

Referenced by operator<<().

operator<=>() [2/2]

```
strong_ordering operator<=> (
            const error_condition & __lhs,
            const error_condition & __rhs)  [related]
```

Ordered comparison for std::error_condition.

This defines a total order by comparing the categories, and then if they are equal comparing the values.

Since

C++11

References error_condition(), and operator<=>().

Referenced by operator<=>().

operator==() [1/3]

```
bool operator== (
            const error_code & __lhs,
            const error_code & __rhs)  [related]
```

Equality comparison for std::error_code.

Returns true only if they have the same category and the same value.

Since

C++11

operator==() [2/3]

```
bool operator== (
            const error_code & __lhs,
            const error_condition & __rhs)  [related]
```

Equality comparison for std::error_code and std::error_condition.

Uses each category's `equivalent` member function to check whether the values correspond to an equivalent error in that category.

Since

C++11

References error_condition().

operator==() [3/3]

```
bool operator== (
            const error_condition & __lhs,
            const error_condition & __rhs)  [related]
```

Equality comparison for std::error_condition.

Returns true only if they have the same category and the same value.

Since

C++11

References error_condition().

system_category()

```
const error_category & std::system_category ()  [noexcept]
```

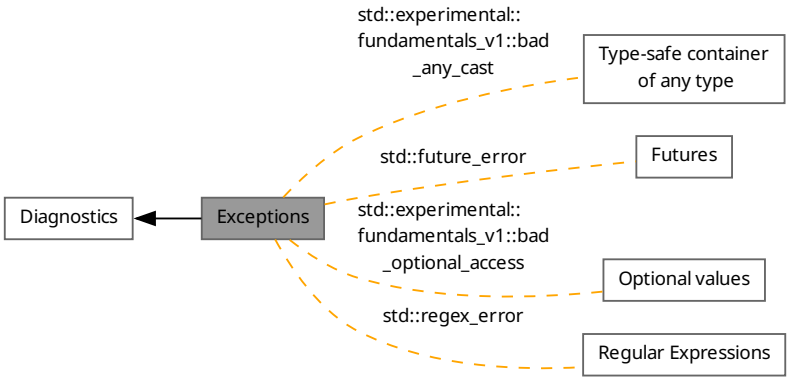
Error category for other error codes defined by the OS.

References make_error_code(), make_error_condition(), and system_category().

Referenced by system_category().

3.5.3 Exceptions

Collaboration diagram for Exceptions:



Classes

- class __cxxabiv1::__forced_unwind
- class std::bad_alloc
- class std::experimental::fundamentals_v1::bad_any_cast
- class std::bad_cast
- class std::bad_exception
- class std::bad_function_call
- class std::experimental::fundamentals_v1::bad_optional_access
- class std::bad_typeid
- class std::bad_weak_ptr
- class std::domain_error
- class std::exception
- class std::__unspecified__::exception_ptr
- class std::ios_base::failure
- struct __gnu_cxx::forced_error
- class std::future_error

- class std::invalid_argument
- class std::length_error
- class std::logic_error
- class std::nested_exception
- class std::out_of_range
- class std::overflow_error
- class std::range_error
- class __gnu_cxx::recursive_init_error
- class std::regex_error
- class std::runtime_error
- class std::system_error
- class std::underflow_error

Typedefs

- typedef void(* std::terminate_handler) ()

Functions

- void __gnu_cxx::__verbose_terminate_handler ()
- exception_ptr std::current_exception () noexcept
- terminate_handler std::get_terminate () noexcept
- template<typename _Ex>
  constexpr exception_ptr std::make_exception_ptr (_Ex __ex) noexcept
- void std::rethrow_exception (exception_ptr)
- template<typename _Ex>
  void std::rethrow_if_nested (const _Ex &__ex)
- terminate_handler std::set_terminate (terminate_handler) noexcept
- void std::terminate () noexcept
- template<typename _Tp>
  void std::throw_with_nested (_Tp &&__t)
- bool std::uncaught_exception () noexcept

3.5.3.1 Detailed Description

Since

C++98

Classes and functions for reporting errors via exceptions.

3.5.3.2 Typedef Documentation

terminate_handler

```
typedef void(* std::terminate_handler) ()
```

If you write a replacement terminate handler, it must be of this type.

3.5.3.3 Function Documentation

__verbose_terminate_handler()

```
void __gnu_cxx::__verbose_terminate_handler ()
```

A replacement for the standard terminate_handler which prints more information about the terminating exception (if any) on stderr.

Call
`std::set_terminate(__gnu_cxx::__verbose_terminate_handler)`

to use. For more info, see http://gcc.gnu.org/onlinedocs/libstdc++/manual/bk01pt02ch06s02.↩
html

In 3.4 and later, this is on by default.

current_exception()

```
exception_ptr std::current_exception ()  [noexcept]
```

Obtain an exception_ptr to the currently handled exception.

If there is none, or the currently handled exception is foreign, return the null value.

Since

C++11

References make_exception_ptr().

Referenced by std::nested_exception::nested_exception(), and make_exception_ptr().

get_terminate()

```
terminate_handler std::get_terminate ()  [noexcept]
```

Return the current terminate handler.

make_exception_ptr()

```
template<typename _Ex>
exception_ptr std::make_exception_ptr (
            _Ex __ex)  [constexpr], [noexcept]
```

Obtain an exception_ptr pointing to a copy of the supplied object.

References current_exception().

Referenced by current_exception().

rethrow_exception()

```
void std::rethrow_exception (
            exception_ptr )
```

Throw the object pointed to by the exception_ptr.

References rethrow_exception().

Referenced by std::__basic_future< _Res & >::_M_get_result(), rethrow_exception(), and std::nested_exception::rethrow_nested().

rethrow_if_nested()

```
template<typename _Ex>
void std::rethrow_if_nested (
            const _Ex & __ex)  [inline]
```

Rethrow a nested exception

If __ex contains a `std::nested_exception` object, call its `rethrow_nested()` member to rethrow the stored exception.

After catching an exception thrown by a call to `std::throw_with_nested` this function can be used to rethrow the exception that was active when `std::throw_with_nested` was called.

Since

C++11

set_terminate()

```
terminate_handler std::set_terminate (
            terminate_handler )  [noexcept]
```

Takes a new handler function as an argument, returns the old function.

terminate()

```
void std::terminate ()  [noexcept]
```

The runtime will call this function if exception handling must be abandoned for any reason. It can also be called by the user.

Referenced by std::nested_exception::rethrow_nested().

throw_with_nested()

```
template<typename _Tp>
void std::throw_with_nested (
             _Tp && __t)  [inline]
```

Throw an exception that also stores the currently active exception.

If _Tp is derived from std::nested_exception or is not usable as a base-class, throws a copy of __t. Otherwise, throws an object of an implementation-defined type derived from both _Tp and std::nested_exception, containing a copy of __t and the result of std::current_exception().

In other words, throws the argument as a new exception that contains the currently active exception nested within it. This is intended for use in a catch handler to replace the caught exception with a different type, while still preserving the original exception. When the new exception is caught, the nested exception can be rethrown by using std::↩ rethrow_if_nested.

This can be used at API boundaries, for example to catch a library's internal exception type and rethrow it nested with a std::runtime_error, or vice versa.

Since

C++11

References forward().

uncaught_exception()

```
bool std::uncaught_exception ()  [noexcept]
```

[18.6.4]/1: 'Returns true after completing evaluation of a throw-expression until either completing initialization of the exception-declaration in the matching handler or entering unexpected() due to the throw; or after entering terminate() for any reason other than an explicit call to terminate(). [Note: This includes stack unwinding [15.2]. end note]'

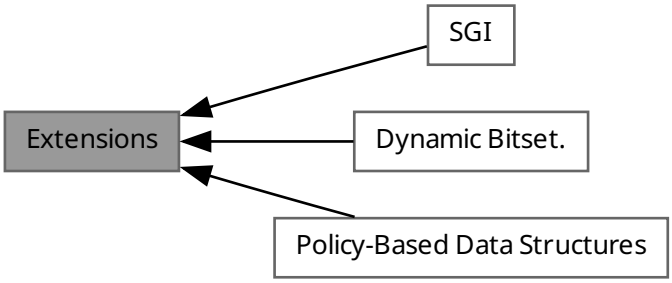
2: 'When uncaught_exception() is true, throwing an exception can result in a call of terminate() (15.5.1).

References uncaught_exception().

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), and uncaught_exception().

3.6 Extensions

Collaboration diagram for Extensions:



Topics

- Dynamic Bitset.
- Policy-Based Data Structures
- SGI

Namespaces

- namespace __gnu_cxx
- namespace std::tr2

Classes

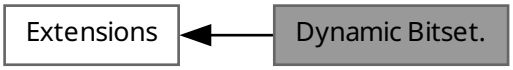
- class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >

3.6.1 Detailed Description

Components generally useful that are not part of any standard.

3.6.2 Dynamic Bitset.

Collaboration diagram for Dynamic Bitset.:



Classes

- struct std::tr2::__dynamic_bitset_base< _WordT, _Alloc >
- class std::tr2::dynamic_bitset< _WordT, _Alloc >

Functions

- template<typename _CharT, typename _Traits, typename _Alloc1>
  void **std::tr2::dynamic_bitset< _WordT, _Alloc >::_M_copy_to_string** (std::basic_string< _CharT, _Traits, ↩ _Alloc1 > &__str, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1')) const
- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
  std::basic_ostream< _CharT, _Traits > & std::tr2::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const dynamic_bitset< _WordT, _Alloc > &__x)
- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
  std::basic_istream< _CharT, _Traits > & std::tr2::operator>> (std::basic_istream< _CharT, _Traits > &__↩ is, dynamic_bitset< _WordT, _Alloc > &__x)

- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator!= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator<= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _↩ Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _↩ Alloc > &__rhs)

- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)

- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)

3.6.2.1 Detailed Description

3.6.2.2 Function Documentation

operator"!=()

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator!= (
            const dynamic_bitset< _WordT, _Alloc > & __lhs,
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

operator&()

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (
            const dynamic_bitset< _WordT, _Alloc > & __x,
            const dynamic_bitset< _WordT, _Alloc > & __y)  [inline]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

Returns

A new bitset.

These should be self-explanatory.

operator-()

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (
            const dynamic_bitset< _WordT, _Alloc > & __x,
            const dynamic_bitset< _WordT, _Alloc > & __y)  [inline]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

Returns

A new bitset.

These should be self-explanatory.

operator<<()

```
template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
std::basic_ostream< _CharT, _Traits > & std::tr2::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const dynamic_bitset< _WordT, _Alloc > & __x)  [inline]
```

Stream output operator for dynamic_bitset.

operator<=()

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator<= (
            const dynamic_bitset< _WordT, _Alloc > & __lhs,
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

operator>()

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator> (
            const dynamic_bitset< _WordT, _Alloc > & __lhs,
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

operator>=()

```
template<typename _WordT, typename _Alloc>
bool std::tr2::operator>= (
            const dynamic_bitset< _WordT, _Alloc > & __lhs,
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

These comparisons for equality/inequality are, well, *bitwise*.

operator>>()

```
template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
std::basic_istream< _CharT, _Traits > & std::tr2::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            dynamic_bitset< _WordT, _Alloc > & __x)
```

Stream input operator for dynamic_bitset.

Input will skip whitespace and only accept '0' and '1' characters. The dynamic_bitset will grow as necessary to hold the string of bits.

References std::basic_string< _CharT, _Traits, _Alloc >::empty(), std::basic_string< _CharT, _Traits, _Alloc >::push_back(), std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_string< _CharT, _Traits, _Alloc >::reserve(), std::tr2::dynamic_bitset< _WordT, _Al std::basic_ios< _CharT, _Traits >::setstate(), std::basic_string< _CharT, _Traits, _Alloc >::size(), std::tr2::dynamic_bitset< _WordT, _All and std::basic_ios< _CharT, _Traits >::widen().

operator^()

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (
            const dynamic_bitset< _WordT, _Alloc > & __x,
            const dynamic_bitset< _WordT, _Alloc > & __y)  [inline]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

Returns

A new bitset.

These should be self-explanatory.

operator"" | ()

```
template<typename _WordT, typename _Alloc>
dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (
            const dynamic_bitset< _WordT, _Alloc > & __x,
            const dynamic_bitset< _WordT, _Alloc > & __y)  [inline]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

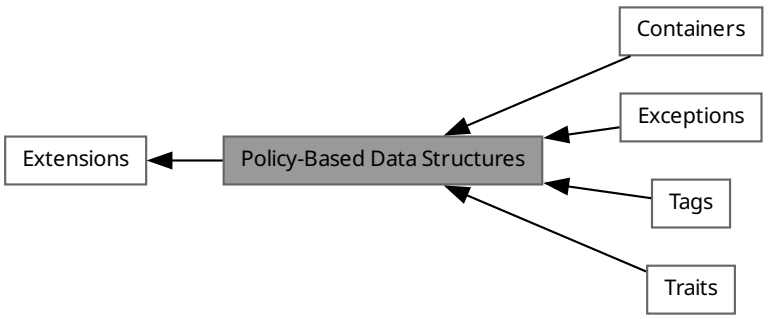
Returns

A new bitset.

These should be self-explanatory.

3.6.3 Policy-Based Data Structures

Collaboration diagram for Policy-Based Data Structures:



Topics

- Containers
- Exceptions
- Tags
- Traits

Classes

- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >

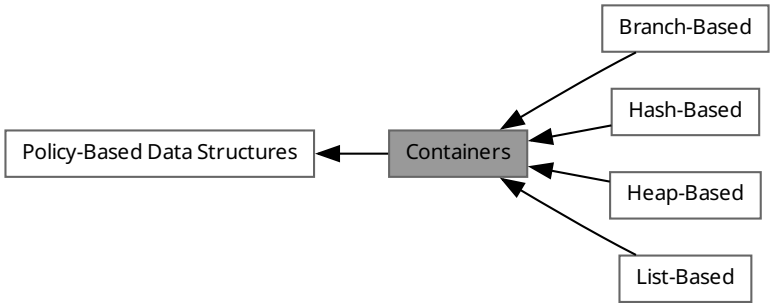
3.6.3.1 Detailed Description

This is a library of policy-based elementary data structures: associative containers and priority queues. It is designed for high-performance, flexibility, semantic safety, and conformance to the corresponding containers in std (except for some points where it differs by design).

For details, see: http://gcc.gnu.org/onlinedocs/libstdc++/ext/pb_ds/index.html

3.6.3.2 Containers

Collaboration diagram for Containers:



Topics

- Branch-Based
- Hash-Based
- Heap-Based
- List-Based

3.6.3.2.1 Detailed Description

3.6.3.2.2 Branch-Based

Collaboration diagram for Branch-Based:



Topics

- Base and Policy Classes

Classes

- class __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >
- class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >
- class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >

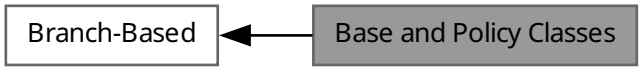
Macros

- #define **PB_DS_BRANCH_BASE**
- #define **PB_DS_TREE_BASE**
- #define **PB_DS_TREE_NODE_AND_IT_TRAITS**
- #define **PB_DS_TRIE_BASE**
- #define **PB_DS_TRIE_NODE_AND_IT_TRAITS**

Detailed Description

Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



*Classes

- class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
- class __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >
- class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >
- class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Detailed Description

3.6.3.2.3 Hash-Based

Collaboration diagram for Hash-Based:



Topics

- Base and Policy Classes

Classes

- class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >
- class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_↩ Hash, _Alloc >
- class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_↩ Policy, Store_Hash, _Alloc >

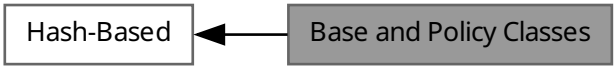
Macros

- #define **PB_DS_CC_HASH_BASE**
- #define **PB_DS_GP_HASH_BASE**
- #define **PB_DS_HASH_BASE**

Detailed Description

Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



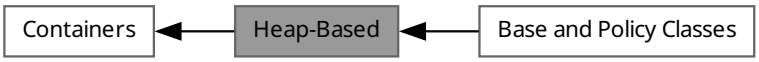
*Classes

- class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >
- class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >

Detailed Description

3.6.3.2.4 Heap-Based

Collaboration diagram for Heap-Based:



Topics

- Base and Policy Classes

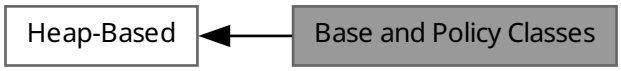
Classes

- class __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >

Detailed Description

Base and Policy Classes

Collaboration diagram for Base and Policy Classes:



*Classes

- class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >
- class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >
- class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >
- class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >
- class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >

Detailed Description

3.6.3.2.5 List-Based

Collaboration diagram for List-Based:



Classes

- class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >

Macros

- #define **PB_DS_LU_BASE**

Detailed Description

3.6.3.3 Exceptions

Collaboration diagram for Exceptions:



Classes

- struct __gnu_pbds::container_error
- struct __gnu_pbds::insert_error
- struct __gnu_pbds::join_error
- struct __gnu_pbds::resize_error

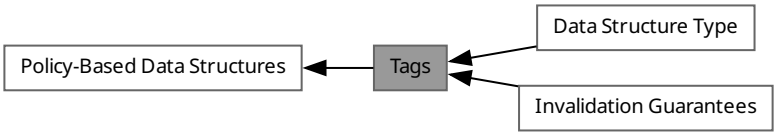
Functions

- void **__gnu_pbds::__throw_container_error** ()
- void **__gnu_pbds::__throw_insert_error** ()
- void **__gnu_pbds::__throw_join_error** ()
- void **__gnu_pbds::__throw_resize_error** ()

3.6.3.3.1 Detailed Description

3.6.3.4 Tags

Collaboration diagram for Tags:



Topics

- Data Structure Type
- Invalidation Guarantees

Classes

- struct __gnu_pbds::trivial_iterator_tag

Typedefs

- typedef void __gnu_pbds::trivial_iterator_difference_type

3.6.3.4.1 Detailed Description

3.6.3.4.2 Typedef Documentation

trivial_iterator_difference_type

```
typedef void __gnu_pbds::trivial_iterator_difference_type
```

Prohibit moving trivial iterators.

3.6.3.4.3 Data Structure Type

Collaboration diagram for Data Structure Type:



Classes

- struct __gnu_pbds::associative_tag
- struct __gnu_pbds::basic_branch_tag
- struct __gnu_pbds::basic_hash_tag
- struct __gnu_pbds::binary_heap_tag
- struct __gnu_pbds::binomial_heap_tag
- struct __gnu_pbds::cc_hash_tag
- struct __gnu_pbds::container_tag
- struct __gnu_pbds::gp_hash_tag
- struct __gnu_pbds::list_update_tag
- struct __gnu_pbds::ov_tree_tag
- struct __gnu_pbds::pairing_heap_tag
- struct __gnu_pbds::pat_trie_tag
- struct __gnu_pbds::priority_queue_tag
- struct __gnu_pbds::rb_tree_tag
- struct __gnu_pbds::rc_binomial_heap_tag
- struct __gnu_pbds::sequence_tag
- struct __gnu_pbds::splay_tree_tag
- struct __gnu_pbds::string_tag
- struct __gnu_pbds::thin_heap_tag
- struct __gnu_pbds::tree_tag
- struct __gnu_pbds::trie_tag

Detailed Description

3.6.3.4.4 Invalidation Guarantees

Collaboration diagram for Invalidation Guarantees:



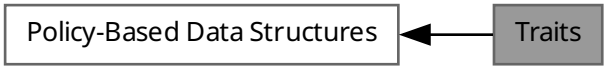
Classes

- struct __gnu_pbds::basic_invalidation_guarantee
- struct __gnu_pbds::point_invalidation_guarantee
- struct __gnu_pbds::range_invalidation_guarantee

Detailed Description

3.6.3.5 Traits

Collaboration diagram for Traits:



Classes

- struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >
- struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >
- struct __gnu_pbds::container_traits< Cntnr >
- struct __gnu_pbds::container_traits_base< _Tag >
- struct __gnu_pbds::container_traits_base< binary_heap_tag >
- struct __gnu_pbds::container_traits_base< binomial_heap_tag >
- struct __gnu_pbds::container_traits_base< cc_hash_tag >
- struct __gnu_pbds::container_traits_base< gp_hash_tag >
- struct __gnu_pbds::container_traits_base< list_update_tag >
- struct __gnu_pbds::container_traits_base< ov_tree_tag >
- struct __gnu_pbds::container_traits_base< pairing_heap_tag >
- struct __gnu_pbds::container_traits_base< pat_trie_tag >
- struct __gnu_pbds::container_traits_base< rb_tree_tag >
- struct __gnu_pbds::container_traits_base< rc_binomial_heap_tag >
- struct __gnu_pbds::container_traits_base< splay_tree_tag >
- struct __gnu_pbds::container_traits_base< thin_heap_tag >
- struct __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >
- struct __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >
- struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >
- struct __gnu_pbds::detail::no_throw_copies< Key, null_type >
- struct __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >
- struct __gnu_pbds::null_type
- struct __gnu_pbds::detail::rebind_traits< _Alloc, T >
- struct __gnu_pbds::detail::select_value_type< Key, Mapped >

- struct __gnu_pbds::detail::select_value_type< Key, null_type >
- struct __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >
- struct __gnu_pbds::detail::stored_data< _Tv, _Th, false >
- struct __gnu_pbds::detail::stored_hash< _Th >
- struct __gnu_pbds::detail::stored_value< _Tv >
- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >
- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >
- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >
- struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >
- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >
- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >
- struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >
- struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >
- struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >
- struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >

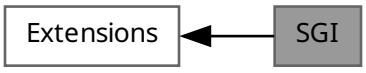
Variables

- template<typename Key, typename _Alloc, bool Store_Hash>
  null_type **__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >::s_null_type**

3.6.3.5.1 Detailed Description

3.6.4 SGI

Collaboration diagram for SGI:



Classes

- class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >
- struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >
- struct __gnu_cxx::constant_unary_fun< _Result, _Argument >
- struct __gnu_cxx::constant_void_fun< _Result >
- class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >
- struct __gnu_cxx::project1st< _Arg1, _Arg2 >
- struct __gnu_cxx::project2nd< _Arg1, _Arg2 >
- struct __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >
- class __gnu_cxx::rope< _CharT, _Alloc >
- struct __gnu_cxx::select1st< _Pair >
- struct __gnu_cxx::select2nd< _Pair >
- class __gnu_cxx::slist< _Tp, _Alloc >
- class __gnu_cxx::subtractive_rng
- struct __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >
- class __gnu_cxx::unary_compose< _Operation1, _Operation2 >

Functions

- template<typename _Tp>
  const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- template<typename _Tp, typename _Compare>
  const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- constexpr size_t std::bitset< _Nb >::_Find_first () const noexcept
- constexpr size_t std::bitset< _Nb >::_Find_next (size_t __prev) const noexcept
- template<class _Operation1, class _Operation2>
  unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1 (const _Operation1 &__fn1, const _Operation2 &__fn2)
- template<class _Operation1, class _Operation2, class _Operation3>
  binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_cxx::compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)
- template<class _Result>
  constant_void_fun< _Result > __gnu_cxx::constant0 (const _Result &__val)
- template<class _Result>
  constant_unary_fun< _Result, _Result > __gnu_cxx::constant1 (const _Result &__val)
- template<class _Result>
  constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2 (const _Result &__val)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  std::pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n (_InputIterator __first, _Size __count, _OutputIterator __result)
- template<typename _InputIterator, typename _Distance>
  void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)
- template<class _Tp>
  _Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)
- template<class _Tp>
  _Tp __gnu_cxx::identity_element (std::plus< _Tp >)

- template<typename _InputIterator1, typename _InputIterator2>
  int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _Tp, typename _Integer>
  _Tp __gnu_cxx::power (_Tp __x, _Integer __n)
- template<typename _Tp, typename _Integer, typename _MonoidOperation>
  _Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _InputIterator, typename _RandomAccessIterator>
  _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random↩ AccessIterator __out_first, _RandomAccessIterator __out_last)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>
  _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random↩ AccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>
  _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator>
  _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)
- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter __first, _Size __count, _↩ ForwardIter __result)

- constexpr bitset< _Nb > & std::bitset< _Nb >::_Unchecked_set (size_t __pos) noexcept
- constexpr bitset< _Nb > & std::bitset< _Nb >::_Unchecked_set (size_t __pos, int __val) noexcept
- constexpr bitset< _Nb > & std::bitset< _Nb >::_Unchecked_reset (size_t __pos) noexcept
- constexpr bitset< _Nb > & std::bitset< _Nb >::_Unchecked_flip (size_t __pos) noexcept
- constexpr bool std::bitset< _Nb >::_Unchecked_test (size_t __pos) const noexcept

3.6.4.1 Detailed Description

Because libstdc++ based its implementation of the STL subsections of the library on the SGI 3.3 implementation, we inherited their extensions as well.

They are additionally documented in the `online documentation`, a copy of which is also shipped with the library source code (in .../docs/html/documentation.html). You can also read the documentation `on SGI's site`, which is still running even though the code is not maintained.

NB that the following notes are pulled from various comments all over the place, so they may seem stilted.
The `identity_element` functions are not part of the C++ standard; SGI provided them as an extension. Its argument is an operation, and its return value is the identity element for that operation. It is overloaded for addition and multiplication, and you can overload it for your own nefarious operations.
As an extension to the binders, SGI provided composition functors and wrapper functions to aid in their creation. The `unary_compose` functor is constructed from two functions/functors, `f` and `g`. Calling `operator()` with a single argument `x` returns `f(g(x))`. The function `compose1` takes the two functions and constructs a `unary_compose` variable for you.
`binary_compose` is constructed from three functors, `f`, `g1`, and `g2`. Its `operator()` returns `f(g1(x),g2(x))`. The function compose2 takes f, g1, and g2, and constructs the `binary_compose` instance for you. For example, if `f` returns an int, then

```
int answer = (compose2(f,g1,g2))(x);
```

is equivalent to

```
int temp1 = g1(x);
int temp2 = g2(x);
int answer = f(temp1,temp2);
```

But the first form is more compact, and can be passed around as a functor to other algorithms.
As an extension, SGI provided a functor called `identity`. When a functor is required but no operations are desired, this can be used as a pass-through. Its `operator()` returns its argument unchanged.
`select1st` and `select2nd` are extensions provided by SGI. Their `operator()s` take a `std::pair` as an argument, and return either the first member or the second member, respectively. They can be used (especially with the composition functors) to *strip* data from a sequence before performing the remainder of an algorithm.
The `operator()` of the `project1st` functor takes two arbitrary arguments and returns the first one, while `project2nd` returns the second one. They are extensions provided by SGI.
These three functors are each constructed from a single arbitrary variable/value. Later, their `operator()s` completely ignore any arguments passed, and return the stored value.

- `constant_void_fun's` `operator()` takes no arguments
- `constant_unary_fun's` `operator()` takes one argument (ignored)
- `constant_binary_fun's` `operator()` takes two arguments (ignored)

The helper creator functions `constant0`, `constant1`, and `constant2` each take a *result* argument and construct variables of the appropriate functor type.

3.6.4.2 Function Documentation

__median() [1/2]

```
template<typename _Tp>
const _Tp & __gnu_cxx::__median (
             const _Tp & __a,
             const _Tp & __b,
             const _Tp & __c)
```

Find the median of three values.

Parameters

_↩ _*a*	A value.
_↩ _*b*	A value.
_↩ _*c*	A value.

Returns

One of `a`, `b` or `c`.

If `{l`,m,n} is some convolution of `{a`,b,c} such that `l<=m<=n` then the value returned will be `m`. This is an SGI extension.

__median() [2/2]

```
template<typename _Tp, typename _Compare>
const _Tp & __gnu_cxx::__median (
             const _Tp & __a,
             const _Tp & __b,
```

```
            const _Tp & __c,
            _Compare __comp)
```

Find the median of three values using a predicate for comparison.

Parameters

__a	A value.
__b	A value.
__c	A value.
__comp	A binary predicate.

Returns

One of `a`, `b` or `c`.

If `{l,m,n}` is some convolution of `{a,b,c}` such that `comp(l,m)` and `comp(m,n)` are both true then the value returned will be `m`. This is an SGI extension.

_Find_first()

```
template<size_t _Nb>
size_t std::bitset< _Nb >::_Find_first () const  [inline], [constexpr], [noexcept]
```

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or size() if not found.

See also

_Find_next

_Find_next()

```
template<size_t _Nb>
size_t std::bitset< _Nb >::_Find_next (
            size_t __prev) const  [inline], [constexpr], [noexcept]
```

Finds the index of the next "on" bit after prev.

Returns

The index of the next bit set, or size() if not found.

Parameters

__prev	Where to start searching.

See also

_Find_first

_Unchecked_flip()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_flip (
            size_t __pos)  [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking. Referenced by flip().

_Unchecked_reset()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_reset (
            size_t __pos)  [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking. Referenced by reset().

_Unchecked_set() [1/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_set (
            size_t __pos)  [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking. Referenced by set().

_Unchecked_set() [2/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::_Unchecked_set (
            size_t __pos,
            int __val)  [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking.

_Unchecked_test()

```
template<size_t _Nb>
bool std::bitset< _Nb >::_Unchecked_test (
            size_t __pos) const  [inline], [constexpr], [noexcept]
```

These versions of single-bit set, reset, flip, and test are extensions from the SGI version. They do no range checking. Referenced by operator[ ](), and test().

compose1()

```
template<class _Operation1, class _Operation2>
unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1 (
            const _Operation1 & __fn1,
            const _Operation2 & __fn2)  [inline]
```

An SGI extension .

compose2()

```
template<class _Operation1, class _Operation2, class _Operation3>
binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_cxx::compose2 (
            const _Operation1 & __fn1,
            const _Operation2 & __fn2,
            const _Operation3 & __fn3)  [inline]
```

An SGI extension .

constant0()

```
template<class _Result>
constant_void_fun< _Result > __gnu_cxx::constant0 (
            const _Result & __val)  [inline]
```

An SGI extension .

constant1()

```
template<class _Result>
constant_unary_fun< _Result, _Result > __gnu_cxx::constant1 (
            const _Result & __val)  [inline]
```

An SGI extension .

constant2()

```
template<class _Result>
constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2 (
            const _Result & __val)  [inline]
```

An SGI extension .

copy_n()

```
template<typename _InputIterator, typename _Size, typename _OutputIterator>
std::pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n (
            _InputIterator __first,
            _Size __count,
            _OutputIterator __result)  [inline]
```

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

Parameters

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

Returns

A std::pair composed of first+count and result+count.

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

distance()

```
template<typename _InputIterator, typename _Distance>
void __gnu_cxx::distance (
            _InputIterator __first,
            _InputIterator __last,
            _Distance & __n)  [inline]
```

This is an SGI extension.

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

References distance().
Referenced by __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::temporary_buffer(), and distance().

identity_element() [1/2]

```
template<class _Tp>
_Tp __gnu_cxx::identity_element (
            std::multiplies< _Tp > )  [inline]
```

An SGI extension .

identity_element() [2/2]

```
template<class _Tp>
_Tp __gnu_cxx::identity_element (
            std::plus< _Tp > )  [inline]
```

An SGI extension .

lexicographical_compare_3way()

```
template<typename _InputIterator1, typename _InputIterator2>
int __gnu_cxx::lexicographical_compare_3way (
            _InputIterator1 __first1,
            _InputIterator1 __last1,
            _InputIterator2 __first2,
            _InputIterator2 __last2)
```

memcmp on steroids.

Parameters

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

Returns

An int, as with memcmp.

The return value will be less than zero if the first range is *lexigraphically less than* the second, greater than zero if the second range is *lexigraphically less than* the first, and zero otherwise. This is an SGI extension.

power() [1/2]

```
template<typename _Tp, typename _Integer>
_Tp __gnu_cxx::power (
            _Tp __x,
            _Integer __n)  [inline]
```

This is an SGI extension.

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

power() [2/2]

```
template<typename _Tp, typename _Integer, typename _MonoidOperation>
_Tp __gnu_cxx::power (
            _Tp __x,
            _Integer __n,
            _MonoidOperation __monoid_op)  [inline]
```

This is an SGI extension.

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

random_sample() [1/2]

```
template<typename _InputIterator, typename _RandomAccessIterator>
_RandomAccessIterator __gnu_cxx::random_sample (
            _InputIterator __first,
            _InputIterator __last,
            _RandomAccessIterator __out_first,
            _RandomAccessIterator __out_last)  [inline]
```

This is an SGI extension.

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

random_sample() [2/2]

```
template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>
_RandomAccessIterator __gnu_cxx::random_sample (
            _InputIterator __first,
            _InputIterator __last,
            _RandomAccessIterator __out_first,
            _RandomAccessIterator __out_last,
            _RandomNumberGenerator & __rand)  [inline]
```

This is an SGI extension.

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

random_sample_n() [1/2]

```
template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>
_OutputIterator __gnu_cxx::random_sample_n (
            _ForwardIterator __first,
            _ForwardIterator __last,
            _OutputIterator __out,
            const _Distance __n)
```

This is an SGI extension.

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

References std::distance(), std::min(), and random_sample_n().
Referenced by random_sample_n().

random_sample_n() [2/2]

```
template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _↩
RandomNumberGenerator>
_OutputIterator __gnu_cxx::random_sample_n (
            _ForwardIterator __first,
            _ForwardIterator __last,
            _OutputIterator __out,
            const _Distance __n,
            _RandomNumberGenerator & __rand)
```

This is an SGI extension.

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

References std::distance(), and std::min().

uninitialized_copy_n()

```
template<typename _InputIter, typename _Size, typename _ForwardIter>
std::pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (
            _InputIter __first,
            _Size __count,
            _ForwardIter __result)  [inline]
```

Copies the range [first,last) into result.

Parameters

__first	An input iterator.
__count	Length
__result	An output iterator.

Returns

__result + (__first + __count)

Like copy(), but does not require an initialized output range.

3.7 Filesystem

Files

- file filesystem

Classes

- class std::filesystem::directory_entry
- class std::filesystem::directory_iterator
- class std::filesystem::file_status
- class std::filesystem::filesystem_error
- class std::filesystem::path::iterator
- class std::filesystem::path
- class std::filesystem::recursive_directory_iterator
- struct std::filesystem::space_info

Typedefs

- using std::filesystem::file_time_type

Enumerations

- enum class std::filesystem::copy_options : unsigned short {
  none , **skip_existing** , **overwrite_existing** , **update_existing** ,
  recursive , **copy_symlinks** , **skip_symlinks** , **directories_only** ,
  create_symlinks , **create_hard_links** }
- enum class std::filesystem::directory_options : unsigned char { **none** , **follow_directory_symlink** , **skip_↩ permission_denied** }
- enum class std::filesystem::file_type : signed char {
  none , **not_found** , **regular** , **directory** ,
  symlink , **block** , **character** , **fifo** ,
  socket , **unknown** }
- enum class std::filesystem::perm_options : unsigned { **replace** , **add** , **remove** , **nofollow** }
- enum class std::filesystem::perms : unsigned {
  none , **owner_read** , **owner_write** , **owner_exec** ,
  owner_all , **group_read** , **group_write** , **group_exec** ,
  group_all , **others_read** , **others_write** , **others_exec** ,
  others_all , **all** , **set_uid** , **set_gid** ,
  sticky_bit , **mask** , **unknown** }

Functions

- path **std::filesystem::absolute** (const path &__p)
- path **std::filesystem::absolute** (const path &__p, error_code &__ec)
- path & **std::filesystem::path::assign** (string_type &&__source)
- iterator **std::filesystem::path::begin** () const noexcept
- path **std::filesystem::canonical** (const path &__p)
- path **std::filesystem::canonical** (const path &__p, error_code &__ec)
- int **std::filesystem::path::compare** (const string_type &__s) const noexcept
- int **std::filesystem::path::compare** (const value_type *__s) const noexcept
- void **std::filesystem::copy** (const path &__from, const path &__to)
- void **std::filesystem::copy** (const path &__from, const path &__to, copy_options __options)
- void **std::filesystem::copy** (const path &__from, const path &__to, copy_options __options, error_code &)
- void **std::filesystem::copy** (const path &__from, const path &__to, error_code &__ec)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option, error_code &)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, error_code &__ec)
- void **std::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_symlink)
- void **std::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_symlink, error_code &__ec) noexcept
- bool **std::filesystem::create_directories** (const path &__p)
- bool **std::filesystem::create_directories** (const path &__p, error_code &__ec)
- bool **std::filesystem::create_directory** (const path &__p)
- bool **std::filesystem::create_directory** (const path &__p, const path &__attributes)
- bool **std::filesystem::create_directory** (const path &__p, const path &__attributes, error_code &__ec) noexcept
- bool **std::filesystem::create_directory** (const path &__p, error_code &__ec) noexcept
- void **std::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink)

- void **std::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- void **std::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link)
- void **std::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link, error_code &__ec) noexcept
- void **std::filesystem::create_symlink** (const path &__to, const path &__new_symlink)
- void **std::filesystem::create_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- path **std::filesystem::current_path** ()
- void **std::filesystem::current_path** (const path &__p)
- void **std::filesystem::current_path** (const path &__p, error_code &__ec) noexcept
- path **std::filesystem::current_path** (error_code &__ec)
- iterator **std::filesystem::path::end** () const noexcept
- bool **std::filesystem::equivalent** (const path &__p1, const path &__p2)
- bool **std::filesystem::equivalent** (const path &__p1, const path &__p2, error_code &__ec) noexcept
- bool **std::filesystem::exists** (const path &__p)
- bool **std::filesystem::exists** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::exists** (file_status) noexcept
- path **std::filesystem::path::extension** () const
- uintmax_t **std::filesystem::file_size** (const path &)
- uintmax_t **std::filesystem::file_size** (const path &, error_code &) noexcept
- path **std::filesystem::path::filename** () const
- std::string **std::filesystem::path::generic_string** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> std::basic_string< _CharT, _Traits, _Allocator > **std::filesystem::path::generic_string** (const _Allocator &__a=_Allocator()) const
- std::u16string **std::filesystem::path::generic_u16string** () const
- std::u32string **std::filesystem::path::generic_u32string** () const
- std::string **std::filesystem::path::generic_u8string** () const
- std::wstring **std::filesystem::path::generic_wstring** () const
- uintmax_t **std::filesystem::hard_link_count** (const path &)
- uintmax_t **std::filesystem::hard_link_count** (const path &, error_code &) noexcept
- bool **std::filesystem::path::has_extension** () const noexcept
- bool **std::filesystem::path::has_stem** () const noexcept
- size_t **std::filesystem::hash_value** (const path &__p) noexcept
- bool **std::filesystem::path::is_absolute** () const noexcept
- bool **std::filesystem::is_block_file** (const path &__p)
- bool **std::filesystem::is_block_file** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_block_file** (file_status __s) noexcept
- bool **std::filesystem::is_character_file** (const path &__p)
- bool **std::filesystem::is_character_file** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_character_file** (file_status __s) noexcept
- bool **std::filesystem::is_directory** (const path &__p)
- bool **std::filesystem::is_directory** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_directory** (file_status __s) noexcept
- bool **std::filesystem::is_empty** (const path &__p)
- bool **std::filesystem::is_empty** (const path &__p, error_code &__ec)
- bool **std::filesystem::is_fifo** (const path &__p)
- bool **std::filesystem::is_fifo** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_fifo** (file_status __s) noexcept
- bool **std::filesystem::is_other** (const path &__p)

- bool **std::filesystem::is_other** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_other** (file_status) noexcept
- bool **std::filesystem::is_regular_file** (const path &__p)
- bool **std::filesystem::is_regular_file** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_regular_file** (file_status) noexcept
- bool **std::filesystem::is_socket** (const path &__p)
- bool **std::filesystem::is_socket** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_socket** (file_status __s) noexcept
- bool **std::filesystem::is_symlink** (const path &__p)
- bool **std::filesystem::is_symlink** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_symlink** (file_status) noexcept
- file_time_type **std::filesystem::last_write_time** (const path &)
- file_time_type **std::filesystem::last_write_time** (const path &, error_code &) noexcept
- void **std::filesystem::last_write_time** (const path &__p, file_time_type __new_time)
- void **std::filesystem::last_write_time** (const path &__p, file_time_type __new_time, error_code &__ec) noexcept
- path & **std::filesystem::path::make_preferred** ()
- copy_options & **std::filesystem::operator&=** (copy_options &__x, copy_options __y) noexcept
- reference **std::filesystem::path::iterator::operator*** () const noexcept
- iterator & **std::filesystem::path::iterator::operator++** () noexcept
- template<typename _CharT>
  __detail::_Path2< _CharT * > & **std::filesystem::path::operator+=** (_CharT __x)
- path & **std::filesystem::path::operator+=** (basic_string_view< value_type > __x)
- path & **std::filesystem::path::operator+=** (const string_type &__x)
- path & **std::filesystem::path::operator+=** (const value_type *__x)
- path & **std::filesystem::path::operator+=** (value_type __x)
- iterator & **std::filesystem::path::iterator::operator--** () noexcept
- path & **std::filesystem::path::operator=** (path &&) noexcept
- path & **std::filesystem::path::operator=** (string_type &&__source)
- constexpr copy_options **std::filesystem::operator^** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::filesystem::operator^=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::filesystem::operator|** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::filesystem::operator|=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::filesystem::operator~** (copy_options __x) noexcept
- void **std::filesystem::permissions** (const path &, perms, perm_options, error_code &) noexcept
- void **std::filesystem::permissions** (const path &__p, perms __prms, error_code &__ec) noexcept
- void **std::filesystem::permissions** (const path &__p, perms __prms, perm_options __opts=perm_options::↩ replace)
- path **std::filesystem::proximate** (const path &__p, const path &__base, error_code &__ec)
- path **std::filesystem::proximate** (const path &__p, const path &__base=current_path())
- path **std::filesystem::proximate** (const path &__p, error_code &__ec)
- path **std::filesystem::read_symlink** (const path &__p)
- path **std::filesystem::read_symlink** (const path &__p, error_code &__ec)
- path **std::filesystem::relative** (const path &__p, const path &__base, error_code &__ec)
- path **std::filesystem::relative** (const path &__p, const path &__base=current_path())
- path **std::filesystem::relative** (const path &__p, error_code &__ec)
- bool **std::filesystem::remove** (const path &, error_code &) noexcept
- bool **std::filesystem::remove** (const path &__p)
- uintmax_t **std::filesystem::remove_all** (const path &)
- uintmax_t **std::filesystem::remove_all** (const path &, error_code &)
- void **std::filesystem::rename** (const path &__from, const path &__to)

- void **std::filesystem::rename** (const path &__from, const path &__to, error_code &__ec) noexcept
- void **std::filesystem::resize_file** (const path &__p, uintmax_t __size)
- void **std::filesystem::resize_file** (const path &__p, uintmax_t __size, error_code &__ec) noexcept
- space_info **std::filesystem::space** (const path &__p)
- space_info **std::filesystem::space** (const path &__p, error_code &__ec) noexcept
- file_status **std::filesystem::status** (const path &)
- file_status **std::filesystem::status** (const path &, error_code &) noexcept
- bool **std::filesystem::status_known** (file_status) noexcept
- path **std::filesystem::path::stem** () const
- std::string **std::filesystem::path::string** () const
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **std::filesystem::path::string** (const _Allocator &__a) const
- void **swap** (path &__lhs, path &__rhs) noexcept
- void **std::filesystem::path::swap** (path &__rhs) noexcept
- file_status **std::filesystem::symlink_status** (const path &)
- file_status **std::filesystem::symlink_status** (const path &, error_code &) noexcept
- path **std::filesystem::temp_directory_path** ()
- path **std::filesystem::temp_directory_path** (error_code &__ec)
- std::u16string **std::filesystem::path::u16string** () const
- std::u32string **std::filesystem::path::u32string** () const
- template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename _CharT = __detail::__value_type_↩
  is_char_or_char8_t<_InputIterator>>
  path u8path (_InputIterator __first, _InputIterator __last)
- template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _CharT = __detail::__value_type_is_char_or↩
  _char8_t<_Source>>
  path u8path (const _Source &__source)
- std::string **std::filesystem::path::u8string** () const
- path **std::filesystem::weakly_canonical** (const path &__p)
- path **std::filesystem::weakly_canonical** (const path &__p, error_code &__ec)
- std::wstring **std::filesystem::path::wstring** () const

- constexpr perms **std::filesystem::operator|** (perms __x, perms __y) noexcept
- constexpr perms **std::filesystem::operator^** (perms __x, perms __y) noexcept
- constexpr perms **std::filesystem::operator~** (perms __x) noexcept
- perms & **std::filesystem::operator&=** (perms &__x, perms __y) noexcept
- perms & **std::filesystem::operator|=** (perms &__x, perms __y) noexcept
- perms & **std::filesystem::operator^=** (perms &__x, perms __y) noexcept

- constexpr perm_options **std::filesystem::operator|** (perm_options __x, perm_options __y) noexcept
- constexpr perm_options **std::filesystem::operator^** (perm_options __x, perm_options __y) noexcept
- constexpr perm_options **std::filesystem::operator~** (perm_options __x) noexcept
- perm_options & **std::filesystem::operator&=** (perm_options &__x, perm_options __y) noexcept
- perm_options & **std::filesystem::operator|=** (perm_options &__x, perm_options __y) noexcept
- perm_options & **std::filesystem::operator^=** (perm_options &__x, perm_options __y) noexcept

- constexpr directory_options **std::filesystem::operator|** (directory_options __x, directory_options __y) noexcept
- constexpr directory_options **std::filesystem::operator^** (directory_options __x, directory_options __y) noexcept
- constexpr directory_options **std::filesystem::operator~** (directory_options __x) noexcept
- directory_options & **std::filesystem::operator&=** (directory_options &__x, directory_options __y) noexcept
- directory_options & **std::filesystem::operator|=** (directory_options &__x, directory_options __y) noexcept
- directory_options & **std::filesystem::operator^=** (directory_options &__x, directory_options __y) noexcept

- directory_iterator begin (directory_iterator __iter) noexcept
- directory_iterator end (directory_iterator) noexcept

- recursive_directory_iterator begin (recursive_directory_iterator __iter) noexcept
- recursive_directory_iterator end (recursive_directory_iterator) noexcept

3.7.1 Detailed Description

3.7.2 Typedef Documentation

file_time_type

```
using std::filesystem::file_time_type
```
The type used for file timestamps.

3.7.3 Enumeration Type Documentation

copy_options

```
enum class std::filesystem::copy_options : unsigned short [strong]
```
Bitmask type controlling effects of `filesystem::copy`.

directory_options

```
enum class std::filesystem::directory_options : unsigned char [strong]
```
Bitmask type controlling directory iteration.

file_type

```
enum class std::filesystem::file_type : signed char [strong]
```
Enumerated type representing the type of a file.

perm_options

```
enum class std::filesystem::perm_options : unsigned [strong]
```
Bitmask type controlling changes to permissions.

perms

```
enum class std::filesystem::perms : unsigned [strong]
```
Bitmask type representing file access permissions.

3.7.4 Function Documentation

begin() [1/2]

```
directory_iterator begin (
            directory_iterator __iter) [related]
```
Enable range-based `for` using directory_iterator.

Since

C++17

e.g. `for (auto& entry : std::filesystem::directory_iterator(".")) ...`

begin() [2/2]

```
recursive_directory_iterator begin (
            recursive_directory_iterator __iter) [related]
```
Enable range-based `for` using recursive_directory_iterator.

Since

C++17

e.g. `for (auto& entry : recursive_directory_iterator(".")) ...`

end() [1/2]

```
directory_iterator end (
            directory_iterator )  [related]
```

Return a past-the-end directory_iterator.

end() [2/2]

```
recursive_directory_iterator end (
            recursive_directory_iterator )  [related]
```

Return a past-the-end recursive_directory_iterator.

u8path() [1/2]

```
template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename
_CharT = __detail::__value_type_is_char_or_char8_t<_InputIterator>>
path u8path (
            _InputIterator __first,
            _InputIterator __last)  [related]
```

Create a path from a UTF-8-encoded sequence of char

Since

C++17

References u8path().
Referenced by u8path(), and u8path().

u8path() [2/2]

```
template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _CharT = __↩
detail::__value_type_is_char_or_char8_t<_Source>>
path u8path (
            const _Source & __source)  [related]
```

Create a path from a UTF-8-encoded sequence of char

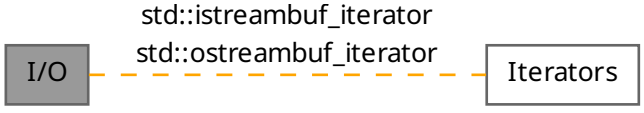
Since

C++17

References u8path().

3.8 I/O

Collaboration diagram for I/O:



Classes

- class std::basic_filebuf< _CharT, _Traits >
- class std::basic_fstream< _CharT, _Traits >
- class std::basic_ifstream< _CharT, _Traits >
- class std::basic_ios< _CharT, _Traits >
- class std::basic_iostream< _CharT, _Traits >
- class std::basic_istream< _CharT, _Traits >
- class std::basic_istringstream< _CharT, _Traits, _Alloc >
- class std::basic_ofstream< _CharT, _Traits >
- class std::basic_ostream< _CharT, _Traits >
- class std::basic_ostringstream< _CharT, _Traits, _Alloc >
- class std::basic_streambuf< _CharT, _Traits >
- class std::basic_stringbuf< _CharT, _Traits, _Alloc >
- class std::basic_stringstream< _CharT, _Traits, _Alloc >
- class std::ios_base
- class std::istreambuf_iterator< _CharT, _Traits >
- class std::ostreambuf_iterator< _CharT, _Traits >
- class __gnu_cxx::stdio_filebuf< _CharT, _Traits >
- class __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >

Typedefs

- typedef basic_filebuf< char > std::filebuf
- typedef basic_fstream< char > std::fstream
- typedef basic_ifstream< char > std::ifstream
- typedef basic_ios< char > std::ios
- typedef basic_iostream< char > std::iostream
- typedef basic_istream< char > std::istream
- typedef basic_istringstream< char > std::istringstream
- typedef basic_ofstream< char > std::ofstream
- typedef basic_ostream< char > std::ostream
- typedef basic_ostringstream< char > std::ostringstream
- typedef basic_streambuf< char > std::streambuf
- typedef basic_stringbuf< char > std::stringbuf
- typedef basic_stringstream< char > std::stringstream
- typedef basic_filebuf< wchar_t > std::wfilebuf
- typedef basic_fstream< wchar_t > std::wfstream
- typedef basic_ifstream< wchar_t > std::wifstream
- typedef basic_ios< wchar_t > std::wios
- typedef basic_iostream< wchar_t > std::wiostream
- typedef basic_istream< wchar_t > std::wistream
- typedef basic_istringstream< wchar_t > std::wistringstream
- typedef basic_ofstream< wchar_t > std::wofstream
- typedef basic_ostream< wchar_t > std::wostream
- typedef basic_ostringstream< wchar_t > std::wostringstream
- typedef basic_streambuf< wchar_t > std::wstreambuf
- typedef basic_stringbuf< wchar_t > std::wstringbuf
- typedef basic_stringstream< wchar_t > std::wstringstream

3.8.1 Detailed Description

Nearly all of the I/O classes are parameterized on the type of characters they read and write. (The major exception is `ios_base` at the top of the hierarchy.) This is a change from pre-Standard streams, which were not templates.
For ease of use and compatibility, all of the `basic_*` I/O-related classes are given typedef names for both of the builtin character widths (wide and narrow). The typedefs are the same as the pre-Standard names, for example:

```
typedef basic_ifstream<char>  ifstream;
```

Because properly forward-declaring these classes can be difficult, you should not do it yourself. Instead, include the `<iosfwd>` header, which contains only declarations of all the I/O classes as well as the typedefs. Trying to forward-declare the typedefs themselves (e.g., `class ostream;`) is not valid ISO C++.
For more specific declarations, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/io.↩html#std.io.objects

3.8.2 Typedef Documentation

filebuf

`typedef basic_filebuf<char> std::filebuf`
Class for `char` file buffers.

fstream

`typedef basic_fstream<char> std::fstream`
Class for `char` mixed input and output file streams.

ifstream

`typedef basic_ifstream<char> std::ifstream`
Class for `char` input file streams.

ios

`typedef basic_ios<char> std::ios`
Base class for `char` streams.

iostream

`typedef basic_iostream<char> std::iostream`
Base class for `char` mixed input and output streams.

istream

`typedef basic_istream<char> std::istream`
Base class for `char` input streams.

istringstream

`typedef basic_istringstream<char> std::istringstream`
Class for `char` input memory streams.

ofstream

`typedef basic_ofstream<char> std::ofstream`
Class for `char` output file streams.

ostream

```
typedef basic_ostream<char> std::ostream
```
Base class for `char` output streams.

ostringstream

```
typedef basic_ostringstream<char> std::ostringstream
```
Class for `char` output memory streams.

streambuf

```
typedef basic_streambuf<char> std::streambuf
```
Base class for `char` buffers.

stringbuf

```
typedef basic_stringbuf<char> std::stringbuf
```
Class for `char` memory buffers.

stringstream

```
typedef basic_stringstream<char> std::stringstream
```
Class for `char` mixed input and output memory streams.

wfilebuf

```
typedef basic_filebuf<wchar_t> std::wfilebuf
```
Class for `wchar_t` file buffers.

wfstream

```
typedef basic_fstream<wchar_t> std::wfstream
```
Class for `wchar_t` mixed input and output file streams.

wifstream

```
typedef basic_ifstream<wchar_t> std::wifstream
```
Class for `wchar_t` input file streams.

wios

```
typedef basic_ios<wchar_t> std::wios
```
Base class for `wchar_t` streams.

wiostream

```
typedef basic_iostream<wchar_t> std::wiostream
```
Base class for `wchar_t` mixed input and output streams.

wistream

```
typedef basic_istream<wchar_t> std::wistream
```
Base class for `wchar_t` input streams.

wistringstream

typedef basic_istringstream<wchar_t> std::wistringstream
Class for wchar_t input memory streams.

wofstream

typedef basic_ofstream<wchar_t> std::wofstream
Class for wchar_t output file streams.

wostream

typedef basic_ostream<wchar_t> std::wostream
Base class for wchar_t output streams.

wostringstream

typedef basic_ostringstream<wchar_t> std::wostringstream
Class for wchar_t output memory streams.

wstreambuf

typedef basic_streambuf<wchar_t> std::wstreambuf
Base class for wchar_t buffers.

wstringbuf

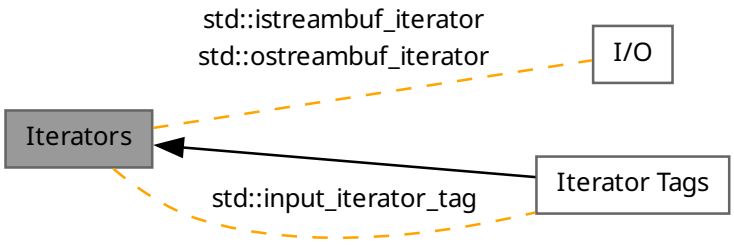
typedef basic_stringbuf<wchar_t> std::wstringbuf
Class for wchar_t memory buffers.

wstringstream

typedef basic_stringstream<wchar_t> std::wstringstream
Class for wchar_t mixed input and output memory streams.

3.9 Iterators

Collaboration diagram for Iterators:



Topics

- Iterator Tags

Classes

- class std::back_insert_iterator< _Container >
- class std::common_iterator< _It, _Sent >
- class std::counted_iterator< _It >
- class std::front_insert_iterator< _Container >
- struct std::input_iterator_tag
- class std::insert_iterator< _Container >
- class std::istream_iterator< _Tp, _CharT, _Traits, _Dist >
- class std::istreambuf_iterator< _CharT, _Traits >
- struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >
- struct std::iterator_traits< _Iterator >
- struct std::iterator_traits< _Tp * >
- class std::move_iterator< _Iterator >
- class std::move_sentinel< _Sent >
- class std::ostream_iterator< _Tp, _CharT, _Traits >
- class std::ostreambuf_iterator< _CharT, _Traits >
- class std::reverse_iterator< _Iterator >

Macros

- #define **_GLIBCXX_MAKE_MOVE_IF_NOEXCEPT_ITERATOR**(_Iter)
- #define **_GLIBCXX_MAKE_MOVE_ITERATOR**(_Iter)

Typedefs

- template<input_iterator _It>
  using **std::const_iterator**
- template<semiregular _Sent>
  using **std::const_sentinel**
- template<indirectly_readable _It>
  using **std::iter_const_reference_t**

Functions

- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::__copy_move_a2** (_CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::__copy_move_a2** (const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT *__result)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **std::__copy_n_a** (istreambuf_iterator< _CharT > __it, _Size __n, _CharT *__result, bool __strict)
- template<typename _Iterator, typename _ReturnType = __conditional_t<__move_if_noexcept_cond <typename iterator_traits<_Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>>
  constexpr _ReturnType **std::__make_move_if_noexcept_iterator** (_Iterator __i)

- template<typename _Tp, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_iterator<_Tp*>>>
  constexpr _ReturnType **std::__make_move_if_noexcept_iterator** (_Tp *__i)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > **std::__make_reverse_iterator** (_Iterator __i)
- template<typename _CharT, typename _Distance>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, void >::__type **std::advance** (istreambuf_iterator< _CharT > &__i, _Distance __n)
- template<typename _Container>
  constexpr back_insert_iterator< _Container > std::back_inserter (_Container &__x)
- template<typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::copy** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)
- template<typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type **std::find** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)
- template<typename _Container>
  constexpr front_insert_iterator< _Container > std::front_inserter (_Container &__x)
- template<typename _Container>
  constexpr insert_iterator< _Container > std::inserter (_Container &__x, std::__detail::__range_iter_t< _Container > __i)
- template<input_iterator _It>
  constexpr const_iterator< _It > **std::make_const_iterator** (_It __it) noexcept(is_nothrow_convertible_v< _It, const_iterator< _It > >)
- template<semiregular _Sent>
  constexpr const_sentinel< _Sent > **std::make_const_sentinel** (_Sent __s) noexcept(is_nothrow_convertible_v< _Sent, const_sentinel< _Sent > >)
- template<typename _Iterator>
  constexpr move_iterator< _Iterator > **std::make_move_iterator** (_Iterator __i)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > std::make_reverse_iterator (_Iterator __i)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() != __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator!=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Iterator>
  requires requires { { __x.base() + __n } -> same_as<_Iterator>; }
  constexpr move_iterator< _Iterator > **std::operator+** (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > &__x)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > **std::operator+** (typename reverse_iterator< _Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)
- template<typename _IteratorL, typename _IteratorR>
  constexpr auto **std::operator-** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) -> decltype(__x.base() - __y.base())
- template<typename _IteratorL, typename _IteratorR>
  constexpr auto **std::operator-** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) -> decltype(__y.base() - __x.base())
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator<** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)

- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() > __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator<** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator<=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() >= __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator<=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<three_way_comparable _Iterator>
  constexpr compare_three_way_result_t< _Iterator > **std::operator<=>** (const move_iterator< _Iterator > &↩ __x, const move_iterator< _Iterator > &__y)
- template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
  constexpr compare_three_way_result_t< _IteratorL, _IteratorR > **std::operator<=>** (const move_iterator< _↩ IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<three_way_comparable _Iterator>
  constexpr compare_three_way_result_t< _Iterator, _Iterator > **std::operator<=>** (const reverse_iterator< _↩ Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
  constexpr compare_three_way_result_t< _IteratorL, _IteratorR > **std::operator<=>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _CharT, typename _Traits>
  bool **std::operator==** (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)
- template<typename _Iterator>
  constexpr bool **std::operator==** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator==** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator==** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)

- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() <= __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)

Variables

- template<typename _Iterator1, typename _Iterator2>
  constexpr bool **std::disable_sized_sentinel_for< move_iterator< _Iterator1 >, move_iterator< _Iterator2 > >**
- template<typename _Iterator1, typename _Iterator2>
  constexpr bool **std::disable_sized_sentinel_for< reverse_iterator< _Iterator1 >, reverse_iterator< _↩ Iterator2 > >**

3.9.1 Detailed Description

Abstractions for uniform iterating through various underlying types.

3.9.2 Function Documentation

back_inserter()

```
template<typename _Container>
back_insert_iterator< _Container > std::back_inserter (
            _Container & __x)  [inline], [nodiscard], [constexpr]
```

Parameters

_↩ _x	A container of arbitrary type.

Returns

An instance of back_insert_iterator working on __x.

This wrapper function helps in creating back_insert_iterator instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.
Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::format(), std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::format(), regex_replace(), regex_replace(), regex_replace(), and regex_replace().

front_inserter()

```
template<typename _Container>
front_insert_iterator< _Container > std::front_inserter (
            _Container & __x)  [inline], [nodiscard], [constexpr]
```

Parameters

_↩ _x	A container of arbitrary type.

Returns

An instance of front_insert_iterator working on `x`.

This wrapper function helps in creating front_insert_iterator instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

inserter()

```
template<typename _Container>
insert_iterator< _Container > std::inserter (
            _Container & __x,
            std::__detail::__range_iter_t< _Container > __i)  [nodiscard], [constexpr]
```

Parameters

_↩ _x	A container of arbitrary type.
_↩ _i	An iterator into the container.

Returns

An instance of insert_iterator working on `__x`.

This wrapper function helps in creating insert_iterator instances. Typing the name of the iterator requires knowing the precise full type of the container, which can be tedious and impedes generic programming. Using this function lets you take advantage of automatic template parameter deduction, making the compiler match the correct types for you.

make_reverse_iterator()

```
template<typename _Iterator>
reverse_iterator< _Iterator > std::make_reverse_iterator (
            _Iterator __i)  [inline], [constexpr]
```

Generator function for reverse_iterator.

operator==()

```
template<typename _IteratorL, typename _IteratorR>
requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
bool std::operator== (
            const reverse_iterator< _IteratorL > & __x,
            const reverse_iterator< _IteratorR > & __y)  [nodiscard], [constexpr]
```

Parameters

_↩ _x	A reverse_iterator.
_↩ _y	A reverse_iterator.

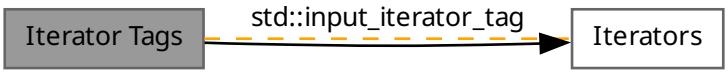
Returns

A simple bool.

Reverse iterators forward comparisons to their underlying base() iterators.
References std::reverse_iterator< _Iterator >::base().

3.9.3 Iterator Tags

Collaboration diagram for Iterator Tags:



Classes

- struct std::bidirectional_iterator_tag
- struct std::contiguous_iterator_tag
- struct std::forward_iterator_tag
- struct std::input_iterator_tag
- struct std::output_iterator_tag
- struct std::random_access_iterator_tag

3.9.3.1 Detailed Description

These are empty types, used to distinguish different iterators. The distinction is not made by what they contain, but simply by what they are. Different underlying algorithms can then be used based on the different operations supported by different iterator types.

3.10 Locales

Classes

- class std::codecvt< _InternT, _ExternT, _StateT >
- class std::ctype< _CharT >
- class std::ctype< char >
- class std::ctype< wchar_t >
- class std::locale::facet
- class std::locale::id
- class std::locale
- class std::messages< _CharT >
- struct std::messages_base
- class std::money_base
- class std::money_get< _CharT, _InIter >
- class std::money_put< _CharT, _OutIter >
- class std::moneypunct< _CharT, _Intl >
- class std::num_get< _CharT, _InIter >

- class std::num_put< _CharT, _OutIter >
- class std::numpunct< _CharT >
- class std::time_base
- class std::time_get< _CharT, _InIter >
- class std::time_put< _CharT, _OutIter >
- class std::wbuffer_convert< _Codecvt, _Elem, _Tr >
- class std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >

Functions

- template<typename _Facet>
  bool std::has_facet (const locale &__loc)
- template<typename _Facet>
  const _Facet & std::use_facet (const locale &__loc)

3.10.1 Detailed Description

Classes and functions for internationalization and localization.

3.10.2 Function Documentation

has_facet()

```
template<typename _Facet>
bool std::has_facet (
            const locale & __loc)  [inline], [nodiscard], [noexcept]
```

Test for the presence of a facet.
has_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Template Parameters

_Facet	The facet type to test the presence of.

Parameters

__loc	The locale to test.

Returns

true if __loc contains a facet of type _Facet, else false.

use_facet()

```
template<typename _Facet>
const _Facet & std::use_facet (
            const locale & __loc)  [inline], [nodiscard]
```

Return a facet.
use_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has_facet(locale) is true, there is a suitable facet to return. It throws std::bad_cast if the locale doesn't contain a facet of type Facet.

Template Parameters

_Facet	The facet type to access.

Parameters

__loc	The locale to use.

Returns

Reference to facet of type Facet.

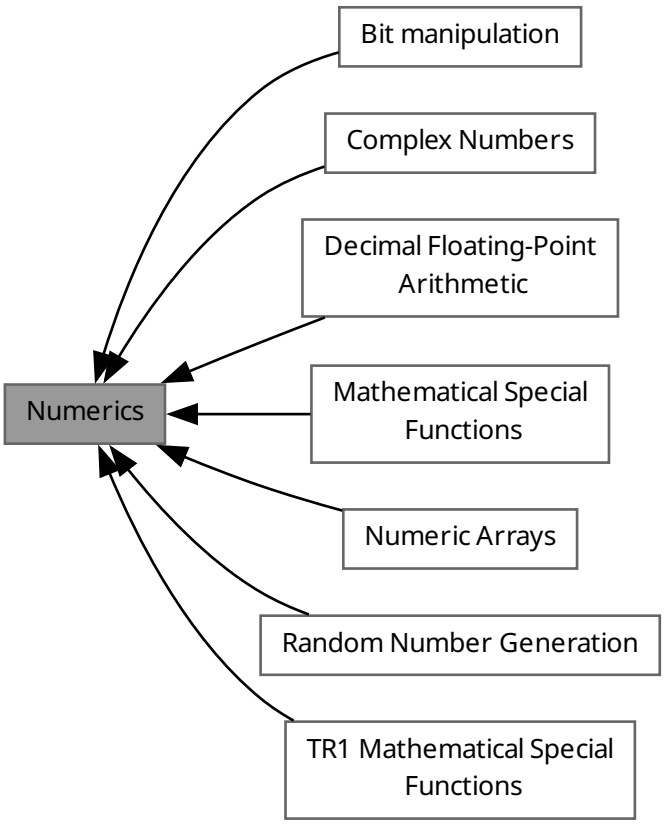
Exceptions

std::bad_cast	if `__loc` doesn't contain a facet of type _Facet.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_year(), std::money_put< _CharT, _OutIter >::do_put(), std::time_put< _CharT, _OutIter >::do
std::time_get< _CharT, _InIter >::get(), isalnum(), isalpha(), isblank(), iscntrl(), std::regex_traits< _Ch_type >::isctype(),
isdigit(), isgraph(), islower(), isprint(), ispunct(), isspace(), isupper(), isxdigit(), std::regex_traits< _Ch_type >::lookup_classname(),
std::regex_traits< _Ch_type >::lookup_collatename(), std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char
std::basic_istream< _CharT, _Traits >::operator>>(), operator>>(), operator>>(), std::time_put< _CharT, _OutIter >::put(),
tolower(), toupper(), std::regex_traits< _CharT >::transform(), std::regex_traits< _CharT >::transform_primary(),
std::regex_traits< _CharT >::translate_nocase(), std::regex_traits< _Ch_type >::value(), and ws().

3.11 Numerics

Collaboration diagram for Numerics:



Topics

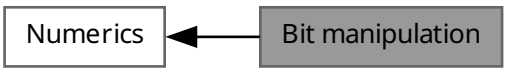
- Bit manipulation
- Complex Numbers
- Decimal Floating-Point Arithmetic
- Mathematical Special Functions
- Numeric Arrays
- Random Number Generation
- TR1 Mathematical Special Functions

3.11.1 Detailed Description

Components for performing numeric operations. Includes support for complex number types, random number generation, numeric (n-at-a-time) arrays, generalized numeric algorithms, and mathematical special functions.

3.11.2 Bit manipulation

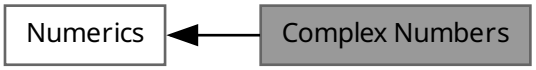
Collaboration diagram for Bit manipulation:



Utilities for examining and manipulating individual bits.

3.11.3 Complex Numbers

Collaboration diagram for Complex Numbers:



Classes

- class std::complex< _Tp >
- class std::complex< double >
- class std::complex< float >
- class std::complex< long double >

Functions

- template<typename _Tp>
  _Tp **std::__complex_abs** (const complex< _Tp > &__z)
- template<typename _Tp>
  _Tp **std::__complex_arg** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_cos** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_cosh** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_exp** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_log** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_pow** (const complex< _Tp > &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  complex< _Tp > **std::__complex_pow_unsigned** (complex< _Tp > __x, unsigned __n)
- template<typename _Tp>
  complex< _Tp > **std::__complex_sin** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_sinh** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_sqrt** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_tan** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_tanh** (const complex< _Tp > &__z)
- template<typename _Tp>
  _Tp std::abs (const complex< _Tp > &)
- template<typename _Tp>
  _Tp std::arg (const complex< _Tp > &)
- template<typename _Tp>
  constexpr complex< _Tp > std::conj (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **std::tr1::conj** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::conj** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > std::cos (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::cosh (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::exp (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)
- template<typename _Tp>
  constexpr _Tp **std::imag** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > std::log (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::log10 (const complex< _Tp > &)
- template<typename _Tp>
  _Tp constexpr std::norm (const complex< _Tp > &)
- template<typename _Tp>
  constexpr complex< _Tp > & **std::complex**< **_Tp** >**::operator*=** (const _Tp &__t)
- template<typename _Up>
  constexpr complex< _Tp > & **std::complex**< **_Tp** >**::operator*=** (const complex< _Up > &__z)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x)
- template<typename _Up>
  constexpr complex< _Tp > & **std::complex**< **_Tp** >**::operator+=** (const complex< _Up > &__z)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x)
- template<typename _Up>
  constexpr complex< _Tp > & **std::complex**< **_Tp** >**::operator-=** (const complex< _Up > &__z)
- template<typename _Tp>
  constexpr complex< _Tp > & **std::complex**< **_Tp** >**::operator/=** (const _Tp &__t)

- template<typename _Up>
  constexpr complex< _Tp > & **std::complex< _Tp >::operator/=** (const complex< _Up > &__z)
- template<typename _Tp, typename _CharT, class _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)
- template ostream & **std::operator**<< (ostream &, const complex< double > &)
- template ostream & **std::operator**<< (ostream &, const complex< float > &)
- template ostream & **std::operator**<< (ostream &, const complex< long double > &)
- template wostream & **std::operator**<< (wostream &, const complex< double > &)
- template wostream & **std::operator**<< (wostream &, const complex< float > &)
- template wostream & **std::operator**<< (wostream &, const complex< long double > &)
- template<typename _Tp>
  constexpr complex< _Tp > & **std::complex< _Tp >::operator=** (const _Tp &__t)
- template<typename _Up>
  constexpr complex< _Tp > & **std::complex< _Tp >::operator=** (const complex< _Up > &__z)
- template<typename _Tp, typename _CharT, class _Traits>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)
- template istream & **std::operator**>> (istream &, complex< double > &)
- template istream & **std::operator**>> (istream &, complex< float > &)
- template istream & **std::operator**>> (istream &, complex< long double > &)
- template wistream & **std::operator**>> (wistream &, complex< double > &)
- template wistream & **std::operator**>> (wistream &, complex< float > &)
- template wistream & **std::operator**>> (wistream &, complex< long double > &)
- template<typename _Tp>
  complex< _Tp > std::polar (const _Tp &, const _Tp &=_Tp(0))
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::tr1::polar** (const _Tp &__rho, const _Up &__theta)
- template<typename _Tp>
  complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)
- template<typename _Tp>
  complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::pow (const complex< _Tp > &, int)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::pow** (const _Tp &__x, const std::complex< _Tp > &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::tr1::pow** (const _Tp &__x, const std::complex< _Up > &__y)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::pow** (const std::complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::pow** (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::tr1::pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)

- template<typename _Tp>
  constexpr _Tp **std::real** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > std::sin (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::sinh (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::sqrt (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::tan (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::tanh (const complex< _Tp > &)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)

3.11.3.1 Detailed Description

Classes and functions for complex numbers.

3.11.3.2 Function Documentation

abs()

```
template<typename _Tp>
_Tp std::abs (
            const complex< _Tp > & __z)  [inline]
```

Return magnitude of *z*.
Referenced by abs(), fabs(), std::tr1::fabs(), std::binomial_distribution< _IntType >::operator()(), and std::poisson_distribution< _IntType >

arg()

```
template<typename _Tp>
_Tp std::arg (
            const complex< _Tp > & __z)  [inline]
```

Return phase angle of *z*.
Referenced by abs().

conj()

```
template<typename _Tp>
complex< _Tp > std::conj (
            const complex< _Tp > & __z)  [inline], [constexpr]
```

Return complex conjugate of *z*.

cos()

```
template<typename _Tp>
complex< _Tp > std::cos (
            const complex< _Tp > & __z)  [inline]
```

Return complex cosine of *z*.
Referenced by polar().

cosh()

```
template<typename _Tp>
complex< _Tp > std::cosh (
            const complex< _Tp > & __z)  [inline]
```

Return complex hyperbolic cosine of *z*.

exp()

```
template<typename _Tp>
complex< _Tp > std::exp (
            const complex< _Tp > & __z)  [inline]
```

Return complex base e exponential of *z*.
Referenced by pow().

fabs()

```
template<typename _Tp>
std::complex< _Tp > std::tr1::fabs (
            const std::complex< _Tp > & __z)  [inline]
```

fabs(__z) [8.1.8].
References std::abs().

log()

```
template<typename _Tp>
complex< _Tp > std::log (
            const complex< _Tp > & __z)  [inline]
```

Return complex natural logarithm of *z*.

Referenced by log10(), std::binomial_distribution< _IntType >::operator()(), std::gamma_distribution< _RealType >::operator()(), std::normal_distribution< _RealType >::operator()(), std::poisson_distribution< _IntType >::operator()(), operator<<(), operator<<(), operator<<(), operator<<(), pow(), and pow().

log10()

```
template<typename _Tp>
complex< _Tp > std::log10 (
            const complex< _Tp > & __z)  [inline]
```

Return complex base 10 logarithm of *z*.

References log().

norm()

```
template<typename _Tp>
_Tp constexpr std::norm (
            const complex< _Tp > & __z)  [inline], [constexpr]
```

Return *z* magnitude squared.

operator*() [1/3]

```
template<typename _Tp>
complex< _Tp > std::operator* (
            const _Tp & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* times *y*.

operator*() [2/3]

```
template<typename _Tp>
complex< _Tp > std::operator* (
            const complex< _Tp > & __x,
            const _Tp & __y)  [inline], [constexpr]
```

Return new complex value *x* times *y*.

operator*() [3/3]

```
template<typename _Tp>
complex< _Tp > std::operator* (
            const complex< _Tp > & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* times *y*.

operator+() [1/4]

```
template<typename _Tp>
complex< _Tp > std::operator+ (
            const _Tp & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* plus *y*.

operator+() [2/4]

```
template<typename _Tp>
complex< _Tp > std::operator+ (
            const complex< _Tp > & __x)  [inline], [constexpr]
```

Return *x*.

operator+() [3/4]

```
template<typename _Tp>
complex< _Tp > std::operator+ (
            const complex< _Tp > & __x,
            const _Tp & __y)  [inline], [constexpr]
```

Return new complex value *x* plus *y*.

operator+() [4/4]

```
template<typename _Tp>
complex< _Tp > std::operator+ (
            const complex< _Tp > & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* plus *y*.

operator-() [1/4]

```
template<typename _Tp>
complex< _Tp > std::operator- (
            const _Tp & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* minus *y*.

operator-() [2/4]

```
template<typename _Tp>
complex< _Tp > std::operator- (
            const complex< _Tp > & __x)  [inline], [constexpr]
```

Return complex negation of *x*.

operator-() [3/4]

```
template<typename _Tp>
complex< _Tp > std::operator- (
            const complex< _Tp > & __x,
            const _Tp & __y)  [inline], [constexpr]
```

Return new complex value *x* minus *y*.

operator-() [4/4]

```
template<typename _Tp>
complex< _Tp > std::operator- (
            const complex< _Tp > & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* minus *y*.

operator/() [1/3]

```
template<typename _Tp>
complex< _Tp > std::operator/ (
            const _Tp & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* divided by *y*.

operator/() [2/3]

```
template<typename _Tp>
complex< _Tp > std::operator/ (
            const complex< _Tp > & __x,
            const _Tp & __y)  [inline], [constexpr]
```

Return new complex value *x* divided by *y*.

operator/() [3/3]

```
template<typename _Tp>
complex< _Tp > std::operator/ (
            const complex< _Tp > & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return new complex value *x* divided by *y*.

operator<<()

```
template<typename _Tp, typename _CharT, class _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const complex< _Tp > & __x)
```

Insertion operator for complex values.
References std::ios_base::failbit, std::basic_istream< _CharT, _Traits >::putback(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_ios< _CharT, _Traits >::widen().

operator==() [1/2]

```
template<typename _Tp>
bool std::operator== (
            const complex< _Tp > & __x,
            const _Tp & __y)  [inline], [constexpr]
```

Return true if *x* is equal to *y*.

operator==() [2/2]

```
template<typename _Tp>
bool std::operator== (
            const complex< _Tp > & __x,
            const complex< _Tp > & __y)  [inline], [constexpr]
```

Return true if *x* is equal to *y*.

operator>>()

```
template<typename _Tp, typename _CharT, class _Traits>
basic_istream< _CharT, _Traits > & std::operator>> (
            basic_istream< _CharT, _Traits > & __is,
            complex< _Tp > & __x)
```

Extraction operator for complex values.

polar()

```
template<typename _Tp>
complex< _Tp > std::polar (
            const _Tp & __rho,
            const _Tp & __theta = _Tp(0))  [inline]
```

Return complex with magnitude *rho* and angle *theta*.
References cos(), and sin().
Referenced by pow(), and pow().

pow() [1/5]

```
template<typename _Tp>
complex< _Tp > std::pow (
            const _Tp & __x,
            const complex< _Tp > & __y)  [inline]
```

Return *x* to the *y'th* power.
References log(), polar(), and pow().

pow() [2/5]

```
template<typename _Tp>
complex< _Tp > std::pow (
            const complex< _Tp > & __x,
            const _Tp & __y)
```

Return *x* to the *y'th* power.
References exp(), log(), polar(), and pow().

pow() [3/5]

```
template<typename _Tp>
complex< _Tp > std::pow (
            const complex< _Tp > & __x,
            const complex< _Tp > & __y)  [inline]
```

Return *x* to the *y'th* power.

pow() [4/5]

```
template<typename _Tp>
complex< _Tp > std::pow (
            const complex< _Tp > & __z,
            int __n)  [inline]
```

Return *x* to the *y'th* power.
Referenced by std::gamma_distribution< _RealType >::operator()(), operator<<(), pow(), pow(), and std::tr1::pow().

pow() [5/5]

```
template<typename _Tp, typename _Up>
std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (
            const std::complex< _Tp > & __x,
            const _Up & __y)  [inline]
```

Additional overloads [8.1.9].
References std::pow().

sin()

```
template<typename _Tp>
complex< _Tp > std::sin (
            const complex< _Tp > & __z)  [inline]
```

Return complex sine of *z*.
Referenced by polar().

sinh()

```
template<typename _Tp>
complex< _Tp > std::sinh (
            const complex< _Tp > & __z)  [inline]
```

Return complex hyperbolic sine of *z*.

sqrt()

```
template<typename _Tp>
complex< _Tp > std::sqrt (
            const complex< _Tp > & __z)  [inline]
```

Return complex square root of *z*.
Referenced by std::normal_distribution< _RealType >::operator()(), and std::student_t_distribution< _RealType >::operator()().

tan()

```
template<typename _Tp>
complex< _Tp > std::tan (
            const complex< _Tp > & __z)  [inline]
```

Return complex tangent of *z*.
Referenced by operator<<().

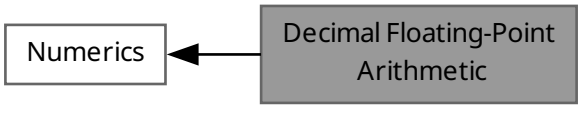
tanh()

```
template<typename _Tp>
complex< _Tp > std::tanh (
            const complex< _Tp > & __z)  [inline]
```

Return complex hyperbolic tangent of *z*.

3.11.4 Decimal Floating-Point Arithmetic

Collaboration diagram for Decimal Floating-Point Arithmetic:



Namespaces

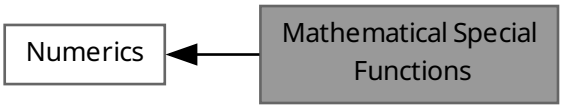
- namespace std::decimal

3.11.4.1 Detailed Description

Classes and functions for decimal floating-point arithmetic.

3.11.5 Mathematical Special Functions

Collaboration diagram for Mathematical Special Functions:



Functions

- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (_Tp __x)
- float __gnu_cxx::airy_aif (float __x)
- long double __gnu_cxx::airy_ail (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (_Tp __x)
- float __gnu_cxx::airy_bif (float __x)
- long double __gnu_cxx::airy_bil (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float std::assoc_laguerref (unsigned int __n, unsigned int __m, float __x)
- long double std::assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float std::assoc_legendref (unsigned int __l, unsigned int __m, float __x)
- long double std::assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)
- template<typename _Tpa, typename _Tpb>
  __gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (_Tpa __a, _Tpb __b)
- float std::betaf (float __a, float __b)
- long double std::betal (long double __a, long double __b)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (_Tp __k)
- float std::comp_ellint_1f (float __k)
- long double std::comp_ellint_1l (long double __k)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (_Tp __k)
- float std::comp_ellint_2f (float __k)
- long double std::comp_ellint_2l (long double __k)

- template<typename _Tp, typename _Tpn>
  __gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 (_Tp __k, _Tpn __nu)
- float std::comp_ellint_3f (float __k, float __nu)
- long double std::comp_ellint_3l (long double __k, long double __nu)
- template<typename _Tpa, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)
- float __gnu_cxx::conf_hypergf (float __a, float __c, float __x)
- long double __gnu_cxx::conf_hypergl (long double __a, long double __c, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float std::cyl_bessel_if (float __nu, float __x)
- long double std::cyl_bessel_il (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float std::cyl_bessel_jf (float __nu, float __x)
- long double std::cyl_bessel_jl (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float std::cyl_bessel_kf (float __nu, float __x)
- long double std::cyl_bessel_kl (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_neumann (_Tpnu __nu, _Tp __x)
- float std::cyl_neumannf (float __nu, float __x)
- long double std::cyl_neumannl (long double __nu, long double __x)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_1 (_Tp __k, _Tpp __phi)
- float std::ellint_1f (float __k, float __phi)
- long double std::ellint_1l (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_2 (_Tp __k, _Tpp __phi)
- float std::ellint_2f (float __k, float __phi)
- long double std::ellint_2l (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp>
  __gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float std::ellint_3f (float __k, float __nu, float __phi)
- long double std::ellint_3l (long double __k, long double __nu, long double __phi)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::expint (_Tp __x)
- float std::expintf (float __x)
- long double std::expintl (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::hermite (unsigned int __n, _Tp __x)
- float std::hermitef (unsigned int __n, float __x)
- long double std::hermitel (unsigned int __n, long double __x)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float __gnu_cxx::hypergf (float __a, float __b, float __c, float __x)
- long double __gnu_cxx::hypergl (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::laguerre (unsigned int __n, _Tp __x)
- float std::laguerref (unsigned int __n, float __x)

- long double std::laguerrel (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::legendre (unsigned int __l, _Tp __x)
- float std::legendref (unsigned int __l, float __x)
- long double std::legendrel (unsigned int __l, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::riemann_zeta (_Tp __s)
- float std::riemann_zetaf (float __s)
- long double std::riemann_zetal (long double __s)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::sph_bessel (unsigned int __n, _Tp __x)
- float std::sph_besself (unsigned int __n, float __x)
- long double std::sph_bessell (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float std::sph_legendref (unsigned int __l, unsigned int __m, float __theta)
- long double std::sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::sph_neumann (unsigned int __n, _Tp __x)
- float std::sph_neumannf (unsigned int __n, float __x)
- long double std::sph_neumannl (unsigned int __n, long double __x)

3.11.5.1 Detailed Description

3.11.5.2 Mathematical Special Functions

A collection of advanced mathematical special functions, defined by ISO/IEC IS 29124 and then added to ISO C++ 2017.

3.11.5.2.1 Introduction and History

The first significant library upgrade on the road to C++2011, TR1, included a set of 23 mathematical functions that significantly extended the standard transcendental functions inherited from C and declared in <cmath>.
Although most components from TR1 were eventually adopted for C++11 these math functions were left behind out of concern for implementability. The math functions were published as a separate international standard IS 29124 - Extensions to the C++ Library to Support Mathematical Special Functions.
For C++17 these functions were incorporated into the main standard.

3.11.5.2.2 Contents

The following functions are implemented in namespace `std`:

- assoc_laguerre - Associated Laguerre functions
- assoc_legendre - Associated Legendre functions
- beta - Beta functions
- comp_ellint_1 - Complete elliptic functions of the first kind
- comp_ellint_2 - Complete elliptic functions of the second kind
- comp_ellint_3 - Complete elliptic functions of the third kind
- cyl_bessel_i - Regular modified cylindrical Bessel functions
- cyl_bessel_j - Cylindrical Bessel functions of the first kind

- cyl_bessel_k - Irregular modified cylindrical Bessel functions
- cyl_neumann - Cylindrical Neumann functions or Cylindrical Bessel functions of the second kind
- ellint_1 - Incomplete elliptic functions of the first kind
- ellint_2 - Incomplete elliptic functions of the second kind
- ellint_3 - Incomplete elliptic functions of the third kind
- expint - The exponential integral
- hermite - Hermite polynomials
- laguerre - Laguerre functions
- legendre - Legendre polynomials
- riemann_zeta - The Riemann zeta function
- sph_bessel - Spherical Bessel functions
- sph_legendre - Spherical Legendre functions
- sph_neumann - Spherical Neumann functions

The hypergeometric functions were stricken from the TR29124 and C++17 versions of this math library because of implementation concerns. However, since they were in the TR1 version and since they are popular we kept them as an extension in namespace `__gnu_cxx`:

- conf_hyperg - Confluent hypergeometric functions
- hyperg - Hypergeometric functions

3.11.5.2.3 Argument Promotion

The arguments suppled to the non-suffixed functions will be promoted according to the following rules:

1. If any argument intended to be floating point is given an integral value That integral value is promoted to double.
2. All floating point arguments are promoted up to the largest floating point precision among them.

3.11.5.2.4 NaN Arguments

If any of the floating point arguments supplied to these functions is invalid or NaN (std::numeric_limits<Tp>::quiet_NaN), the value NaN is returned.

3.11.5.2.5 Implementation

We strive to implement the underlying math with type generic algorithms to the greatest extent possible. In practice, the functions are thin wrappers that dispatch to function templates. Type dependence is controlled with std::numeric_limits and functions thereof.
We don't promote `float` to `double` or `double` to `long double` reflexively. The goal is for `float` functions to operate more quickly, at the cost of `float` accuracy and possibly a smaller domain of validity. Similaryly, `long double` should give you more dynamic range and slightly more pecision than `double` on many systems.

3.11.5.2.6 Testing

These functions have been tested against equivalent implementations from the Gnu Scientific Library, GSL and Boost and the ratio

$$\frac{|f - f_{test}|}{|f_{test}|}$$

is generally found to be within 10^{-15} for 64-bit double on linux-x86_64 systems over most of the ranges of validity.

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

3.11.5.2.7 General Bibliography

See also

Abramowitz and Stegun: Handbook of Mathematical Functions, with Formulas, Graphs, and Mathematical Tables Edited by Milton Abramowitz and Irene A. Stegun, National Bureau of Standards Applied Mathematics Series - 55 Issued June 1964, Tenth Printing, December 1972, with corrections Electronic versions of A&S abound including both pdf and navigable html.

for example http://people.math.sfu.ca/~cbm/aands/

The old A&S has been redone as the NIST Digital Library of Mathematical Functions: http://dlmf.nist.gov/ This version is far more navigable and includes more recent work.

An Atlas of Functions: with Equator, the Atlas Function Calculator 2nd Edition, by Oldham, Keith B., Myland, Jan, Spanier, Jerome

Asymptotics and Special Functions by Frank W. J. Olver, Academic Press, 1974

Numerical Recipes in C, The Art of Scientific Computing, by William H. Press, Second Ed., Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, Cambridge University Press, 1992

The Special Functions and Their Approximations: Volumes 1 and 2, by Yudell L. Luke, Academic Press, 1969

3.11.5.3 Function Documentation

airy_ai()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (
            _Tp __x)  [inline]
```

Return the Airy function $Ai(x)$ of real argument x.

airy_aif()

```
float __gnu_cxx::airy_aif (
            float __x)  [inline]
```

Return the Airy function $Ai(x)$ of `float` argument x.

airy_ail()

```
long double __gnu_cxx::airy_ail (
            long double __x)  [inline]
```

Return the Airy function $Ai(x)$ of `long double` argument x.

airy_bi()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (
            _Tp __x)  [inline]
```

Return the Airy function $Bi(x)$ of real argument x.

airy_bif()

```
float __gnu_cxx::airy_bif (
            float __x)  [inline]
```

Return the Airy function $Bi(x)$ of `float` argument x.

airy_bil()

```
long double __gnu_cxx::airy_bil (
            long double __x)  [inline]
```

Return the Airy function $Bi(x)$ of `long double` argument x.

assoc_laguerre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (
            unsigned int __n,
            unsigned int __m,
            _Tp __x)  [inline]
```

Return the associated Laguerre polynomial of nonnegative order `n`, nonnegative degree `m` and real argument `x`: $L_n^m(x)$.

The associated Laguerre function of real degree α, $L_n^\alpha(x)$, is defined by

$$L_n^\alpha(x) = \frac{(\alpha+1)_n}{n!}\,{}_1F_1(-n;\alpha+1;x)$$

where $(\alpha)_n$ is the Pochhammer symbol and ${}_1F_1(a;c;x)$ is the confluent hypergeometric function.

The associated Laguerre polynomial is defined for integral degree $\alpha = m$ by:

$$L_n^m(x) = (-1)^m \frac{d^m}{dx^m} L_{n+m}(x)$$

where the Laguerre polynomial is defined by:

$$L_n(x) = \frac{e^x}{n!}\frac{d^n}{dx^n}(x^n e^{-x})$$

and $x >= 0$.

See also

laguerre for details of the Laguerre function of degree `n`

Template Parameters

_Tp	The floating-point type of the argument `__x`.

Parameters

_n	The order of the Laguerre function, `__n >= 0`.
_m	The degree of the Laguerre function, `__m >= 0`.

_↩ _x	The argument of the Laguerre function, __x >= 0.

Exceptions

std::domain_error	if __x < 0.

assoc_laguerref()

```
float std::assoc_laguerref (
            unsigned int __n,
            unsigned int __m,
            float __x)  [inline]
```

Return the associated Laguerre polynomial of order n, degree m: $L_n^m(x)$ for float argument.

See also

assoc_laguerre for more details.

assoc_laguerrel()

```
long double std::assoc_laguerrel (
            unsigned int __n,
            unsigned int __m,
            long double __x)  [inline]
```

Return the associated Laguerre polynomial of order n, degree m: $L_n^m(x)$.

See also

assoc_laguerre for more details.

assoc_legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (
            unsigned int __l,
            unsigned int __m,
            _Tp __x)  [inline]
```

Return the associated Legendre function of degree l and order m.
The associated Legendre function is derived from the Legendre function $P_l(x)$ by the Rodrigues formula:

$$P_l^m(x) = (1 - x^2)^{m/2} \frac{d^m}{dx^m} P_l(x)$$

See also

legendre for details of the Legendre function of degree `l`

Template Parameters

_Tp	The floating-point type of the argument `__x`.

Parameters

_↩ _l	The degree `__l >= 0`.
_↩ _m	The order `__m <= l`.
_↩ _x	The argument, `abs(__x) <= 1`.

Exceptions

std::domain_error	if `abs(__x) > 1`.

assoc_legendref()

```
float std::assoc_legendref (
            unsigned int __l,
            unsigned int __m,
            float __x)  [inline]
```

Return the associated Legendre function of degree `l` and order `m` for `float` argument.

See also

assoc_legendre for more details.

assoc_legendrel()

```
long double std::assoc_legendrel (
            unsigned int __l,
            unsigned int __m,
            long double __x)  [inline]
```

Return the associated Legendre function of degree `l` and order `m`.

See also

assoc_legendre for more details.

beta()

```
template<typename _Tpa, typename _Tpb>
__gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (
            _Tpa __a,
            _Tpb __b)  [inline]
```

Return the beta function, $B(a, b)$, for real parameters a, b.
The beta function is defined by

$$B(a,b) = \int_0^1 t^{a-1}(1-t)^{b-1}dt = \frac{\Gamma(a)\Gamma(b)}{\Gamma(a+b)}$$

where $a > 0$ and $b > 0$

Template Parameters

_Tpa	The floating-point type of the parameter __a.
_Tpb	The floating-point type of the parameter __b.

Parameters

_a	The first argument of the beta function, __a > 0 .
_b	The second argument of the beta function, __b > 0 .

Exceptions

std::domain_error	if __a < 0 or __b < 0 .

betaf()

```
float std::betaf (
            float __a,
            float __b)  [inline]
```

Return the beta function, $B(a, b)$, for float parameters a, b.

See also

beta for more details.

betal()

```
long double std::betal (
            long double __a,
            long double __b)  [inline]
```

Return the beta function, $B(a, b)$, for long double parameters a, b.

See also

beta for more details.

comp_ellint_1()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (
            _Tp __k)  [inline]
```

Return the complete elliptic integral of the first kind $K(k)$ for real modulus k.
The complete elliptic integral of the first kind is defined as

$$K(k) = F(k, \pi/2) = \int_0^{\pi/2} \frac{d\theta}{\sqrt{1 - k^2 sin^2\theta}}$$

where $F(k, \phi)$ is the incomplete elliptic integral of the first kind and the modulus $|k| <= 1$.

See also

ellint_1 for details of the incomplete elliptic function of the first kind.

Template Parameters

_Tp	The floating-point type of the modulus __k.

Parameters

_↩ _k	The modulus, abs(__k) <= 1

Exceptions

std::domain_error	if abs(__k) > 1 .

comp_ellint_1f()

```
float std::comp_ellint_1f (
            float __k)  [inline]
```

Return the complete elliptic integral of the first kind $E(k)$ for float modulus k.

See also

comp_ellint_1 for details.

comp_ellint_1l()

```
long double std::comp_ellint_1l (
            long double __k)  [inline]
```

Return the complete elliptic integral of the first kind $E(k)$ for long double modulus k.

See also

comp_ellint_1 for details.

comp_ellint_2()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (
            _Tp __k)  [inline]
```

Return the complete elliptic integral of the second kind $E(k)$ for real modulus k.
The complete elliptic integral of the second kind is defined as

$$E(k) = E(k, \pi/2) = \int_0^{\pi/2} \sqrt{1 - k^2 sin^2\theta}$$

where $E(k, \phi)$ is the incomplete elliptic integral of the second kind and the modulus $|k| <= 1$.

See also

ellint_2 for details of the incomplete elliptic function of the second kind.

Template Parameters

_Tp	The floating-point type of the modulus __k.

Parameters

_↩ _k	The modulus, abs(__k) <= 1

Exceptions

std::domain_error	if abs(__k) > 1.

comp_ellint_2f()

```
float std::comp_ellint_2f (
            float __k)  [inline]
```

Return the complete elliptic integral of the second kind $E(k)$ for float modulus k.

See also

comp_ellint_2 for details.

comp_ellint_2l()

```
long double std::comp_ellint_2l (
            long double __k)  [inline]
```

Return the complete elliptic integral of the second kind $E(k)$ for long double modulus k.

See also

comp_ellint_2 for details.

comp_ellint_3()

```
template<typename _Tp, typename _Tpn>
__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 (
            _Tp __k,
            _Tpn __nu)  [inline]
```

Return the complete elliptic integral of the third kind $\Pi(k,\nu) = \Pi(k,\nu,\pi/2)$ for real modulus k.
The complete elliptic integral of the third kind is defined as

$$\Pi(k,\nu) = \Pi(k,\nu,\pi/2) = \int_0^{\pi/2} \frac{d\theta}{(1-\nu\sin^2\theta)\sqrt{1-k^2\sin^2\theta}}$$

where $\Pi(k,\nu,\phi)$ is the incomplete elliptic integral of the second kind and the modulus $|k| <= 1$.

See also

ellint_3 for details of the incomplete elliptic function of the third kind.

Template Parameters

_Tp	The floating-point type of the modulus __k.
_Tpn	The floating-point type of the argument __nu.

Parameters

__k	The modulus, abs(__k) <= 1
__nu	The argument

Exceptions

std::domain_error	if abs(__k) > 1.

comp_ellint_3f()

```
float std::comp_ellint_3f (
            float __k,
            float __nu)  [inline]
```

Return the complete elliptic integral of the third kind $\Pi(k,\nu)$ for float modulus k.

See also

comp_ellint_3 for details.

comp_ellint_3l()

```
long double std::comp_ellint_3l (
            long double __k,
            long double __nu)  [inline]
```

Return the complete elliptic integral of the third kind $\Pi(k,\nu)$ for long double modulus k.

See also

comp_ellint_3 for details.

conf_hyperg()

```
template<typename _Tpa, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (
            _Tpa __a,
            _Tpc __c,
            _Tp __x)  [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a;c;x)$ of real numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.
The confluent hypergeometric function is defined by

$$ {}_1F_1(a;c;x) = \sum_{n=0}^{\infty} \frac{(a)_n x^n}{(c)_n n!} $$

where the Pochhammer symbol is $(x)_k = (x)(x+1)...(x+k-1)$, $(x)_0 = 1$

Parameters

_↩ _a	The numeratorial parameter
_↩ _c	The denominatorial parameter
_↩ _x	The argument

conf_hypergf()

```
float __gnu_cxx::conf_hypergf (
            float __a,
            float __c,
            float __x)  [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a;c;x)$ of `float` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

See also

conf_hyperg for details.

conf_hypergl()

```
long double __gnu_cxx::conf_hypergl (
            long double __a,
            long double __c,
            long double __x)  [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a;c;x)$ of `long double` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

See also

conf_hyperg for details.

cyl_bessel_i()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i (
            _Tpnu __nu,
            _Tp __x)  [inline]
```

Return the regular modified Bessel function $I_\nu(x)$ for real order ν and argument $x >= 0$.
The regular modified cylindrical Bessel function is:

$$I_\nu(x) = i^{-\nu} J_\nu(ix) = \sum_{k=0}^{\infty} \frac{(x/2)^{\nu+2k}}{k!\Gamma(\nu+k+1)}$$

Template Parameters

_Tpnu	The floating-point type of the order __nu.
_Tp	The floating-point type of the argument __x.

Parameters

__nu	The order
__x	The argument, __x >= 0

Exceptions

std::domain_error	if __x < 0 .

cyl_bessel_if()

```
float std::cyl_bessel_if (
            float __nu,
            float __x)  [inline]
```

Return the regular modified Bessel function $I_\nu(x)$ for `float` order ν and argument $x >= 0$.

See also

cyl_bessel_i for setails.

cyl_bessel_il()

```
long double std::cyl_bessel_il (
            long double __nu,
            long double __x)  [inline]
```

Return the regular modified Bessel function $I_\nu(x)$ for `long double` order ν and argument $x >= 0$.

See also

cyl_bessel_i for setails.

cyl_bessel_j()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j (
            _Tpnu __nu,
            _Tp __x)  [inline]
```

Return the Bessel function $J_\nu(x)$ of real order ν and argument $x >= 0$.
The cylindrical Bessel function is:

$$J_\nu(x) = \sum_{k=0}^{\infty} \frac{(-1)^k (x/2)^{\nu+2k}}{k!\Gamma(\nu+k+1)}$$

Template Parameters

_Tpnu	The floating-point type of the order __nu.
_Tp	The floating-point type of the argument __x.

Parameters

__nu	The order
__x	The argument, __x >= 0

Exceptions

std::domain_error	if __x < 0 .

cyl_bessel_jf()

```
float std::cyl_bessel_jf (
            float __nu,
            float __x)  [inline]
```

Return the Bessel function of the first kind $J_\nu(x)$ for `float` order ν and argument $x >= 0$.

See also

cyl_bessel_j for setails.

cyl_bessel_jl()

```
long double std::cyl_bessel_jl (
            long double __nu,
            long double __x)  [inline]
```

Return the Bessel function of the first kind $J_\nu(x)$ for `long double` order ν and argument $x >= 0$.

See also

cyl_bessel_j for setails.

cyl_bessel_k()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k (
            _Tpnu __nu,
            _Tp __x)  [inline]
```

Return the irregular modified Bessel function $K_\nu(x)$ of real order ν and argument x.
The irregular modified Bessel function is defined by:

$$K_\nu(x) = \frac{\pi}{2}\frac{I_{-\nu}(x) - I_\nu(x)}{\sin \nu\pi}$$

where for integral $\nu = n$ a limit is taken: $lim_{\nu \to n}$. For negative argument we have simply:

$$K_{-\nu}(x) = K_\nu(x)$$

Template Parameters

_Tpnu	The floating-point type of the order __nu.
_Tp	The floating-point type of the argument __x.

Parameters

__nu	The order
__x	The argument, __x >= 0

Exceptions

std::domain_error	if __x < 0 .

cyl_bessel_kf()

```
float std::cyl_bessel_kf (
            float __nu,
            float __x)  [inline]
```

Return the irregular modified Bessel function $K_\nu(x)$ for `float` order ν and argument $x >= 0$.

See also

cyl_bessel_k for setails.

cyl_bessel_kl()

```
long double std::cyl_bessel_kl (
            long double __nu,
            long double __x)  [inline]
```

Return the irregular modified Bessel function $K_\nu(x)$ for `long double` order ν and argument $x >= 0$.

See also

cyl_bessel_k for setails.

cyl_neumann()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_neumann (
            _Tpnu __nu,
            _Tp __x)  [inline]
```

Return the Neumann function $N_\nu(x)$ of real order ν and argument $x >= 0$.
The Neumann function is defined by:

$$N_\nu(x) = \frac{J_\nu(x)\cos\nu\pi - J_{-\nu}(x)}{\sin\nu\pi}$$

where $x >= 0$ and for integral order $\nu = n$ a limit is taken: $lim_{\nu\to n}$.

Template Parameters

_Tpnu	The floating-point type of the order __nu.
_Tp	The floating-point type of the argument __x.

Parameters

__nu	The order
__x	The argument, __x >= 0

Exceptions

std::domain_error	if __x < 0 .

cyl_neumannf()

```
float std::cyl_neumannf (
            float __nu,
            float __x)  [inline]
```

Return the Neumann function $N_\nu(x)$ of `float` order ν and argument x.

See also

cyl_neumann for setails.

cyl_neumannl()

```
long double std::cyl_neumannl (
            long double __nu,
            long double __x)  [inline]
```

Return the Neumann function $N_\nu(x)$ of `long double` order ν and argument x.

See also

cyl_neumann for setails.

ellint_1()

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_1 (
            _Tp __k,
            _Tpp __phi)  [inline]
```

Return the incomplete elliptic integral of the first kind $F(k,\phi)$ for `real` modulus k and angle ϕ.
The incomplete elliptic integral of the first kind is defined as

$$F(k,\phi) = \int_0^\phi \frac{d\theta}{\sqrt{1 - k^2 sin^2\theta}}$$

For $\phi = \pi/2$ this becomes the complete elliptic integral of the first kind, $K(k)$.

See also

comp_ellint_1.

Template Parameters

_Tp	The floating-point type of the modulus `__k`.
_Tpp	The floating-point type of the angle `__phi`.

Parameters

__k	The modulus, `abs(__k) <= 1`
__phi	The integral limit argument in radians

Exceptions

std::domain_error	if `abs(__k) > 1` .

ellint_1f()

```
float std::ellint_1f (
            float __k,
            float __phi)  [inline]
```

Return the incomplete elliptic integral of the first kind $E(k,\phi)$ for `float` modulus k and angle ϕ.

See also

ellint_1 for details.

ellint_1l()

```
long double std::ellint_1l (
            long double __k,
            long double __phi)  [inline]
```

Return the incomplete elliptic integral of the first kind $E(k, \phi)$ for `long double` modulus k and angle ϕ.

See also

ellint_1 for details.

ellint_2()

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_2 (
            _Tp __k,
            _Tpp __phi)  [inline]
```

Return the incomplete elliptic integral of the second kind $E(k, \phi)$.
The incomplete elliptic integral of the second kind is defined as

$$E(k, \phi) = \int_0^\phi \sqrt{1 - k^2 sin^2\theta}$$

For $\phi = \pi/2$ this becomes the complete elliptic integral of the second kind, $E(k)$.

See also

comp_ellint_2.

Template Parameters

_Tp	The floating-point type of the modulus `__k`.
_Tpp	The floating-point type of the angle `__phi`.

Parameters

__k	The modulus, `abs(__k) <= 1`
__phi	The integral limit argument in radians

Returns

The elliptic function of the second kind.

Exceptions

std::domain_error	if `abs(__k) > 1` .

ellint_2f()

```
float std::ellint_2f (
            float __k,
            float __phi)  [inline]
```

Return the incomplete elliptic integral of the second kind $E(k, \phi)$ for `float` argument.

See also

ellint_2 for details.

ellint_2l()

```
long double std::ellint_2l (
            long double __k,
            long double __phi)  [inline]
```

Return the incomplete elliptic integral of the second kind $E(k, \phi)$.

See also

ellint_2 for details.

ellint_3()

```
template<typename _Tp, typename _Tpn, typename _Tpp>
__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::ellint_3 (
            _Tp __k,
            _Tpn __nu,
            _Tpp __phi)  [inline]
```

Return the incomplete elliptic integral of the third kind $\Pi(k, \nu, \phi)$.
The incomplete elliptic integral of the third kind is defined by:

$$\Pi(k, \nu, \phi) = \int_0^\phi \frac{d\theta}{(1 - \nu \sin^2\theta)\sqrt{1 - k^2 \sin^2\theta}}$$

For $\phi = \pi/2$ this becomes the complete elliptic integral of the third kind, $\Pi(k, \nu)$.

See also

comp_ellint_3.

Template Parameters

_Tp	The floating-point type of the modulus `__k`.
_Tpn	The floating-point type of the argument `__nu`.
_Tpp	The floating-point type of the angle `__phi`.

Parameters

__k	The modulus, `abs(__k) <= 1`

__nu	The second argument
__phi	The integral limit argument in radians

Returns

The elliptic function of the third kind.

Exceptions

std::domain_error	if `abs(__k) > 1 .`

ellint_3f()

```
float std::ellint_3f (
            float __k,
            float __nu,
            float __phi)  [inline]
```

Return the incomplete elliptic integral of the third kind $\Pi(k, \nu, \phi)$ for `float` argument.

See also

ellint_3 for details.

ellint_3l()

```
long double std::ellint_3l (
            long double __k,
            long double __nu,
            long double __phi)  [inline]
```

Return the incomplete elliptic integral of the third kind $\Pi(k, \nu, \phi)$.

See also

ellint_3 for details.

expint()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::expint (
            _Tp __x)  [inline]
```

Return the exponential integral $Ei(x)$ for `real` argument `x`.
The exponential integral is given by

$$Ei(x) = -\int_{-x}^{\infty} \frac{e^t}{t} dt$$

Template Parameters

_Tp	The floating-point type of the argument __x.

Parameters

_↩ _x	The argument of the exponential integral function.

expintf()

```
float std::expintf (
            float __x)  [inline]
```

Return the exponential integral $Ei(x)$ for `float` argument `x`.

See also

expint for details.

expintl()

```
long double std::expintl (
            long double __x)  [inline]
```

Return the exponential integral $Ei(x)$ for `long double` argument `x`.

See also

expint for details.

hermite()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::hermite (
            unsigned int __n,
            _Tp __x)  [inline]
```

Return the Hermite polynomial $H_n(x)$ of order n and `real` argument `x`.
The Hermite polynomial is defined by:

$$H_n(x) = (-1)^n e^{x^2} \frac{d^n}{dx^n} e^{-x^2}$$

The Hermite polynomial obeys a reflection formula:

$$H_n(-x) = (-1)^n H_n(x)$$

Template Parameters

_Tp	The floating-point type of the argument __x.

Parameters

_↩ _n	The order
_↩ _x	The argument

hermitef()

```
float std::hermitef (
            unsigned int __n,
            float __x)  [inline]
```

Return the Hermite polynomial $H_n(x)$ of nonnegative order n and float argument x.

See also

hermite for details.

hermitel()

```
long double std::hermitel (
            unsigned int __n,
            long double __x)  [inline]
```

Return the Hermite polynomial $H_n(x)$ of nonnegative order n and long double argument x.

See also

hermite for details.

hyperg()

```
template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg (
            _Tpa __a,
            _Tpb __b,
            _Tpc __c,
            _Tp __x)  [inline]
```

Return the hypergeometric function ${}_2F_1(a,b;c;x)$ of real numeratorial parameters a and b, denominatorial parameter c, and argument x.
The hypergeometric function is defined by

$$ {}_2F_1(a;c;x) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n x^n}{(c)_n n!} $$

where the Pochhammer symbol is $(x)_k = (x)(x+1)...(x+k-1)$, $(x)_0 = 1$

Parameters

_↩ _a	The first numeratorial parameter
_↩ _b	The second numeratorial parameter
_↩ _c	The denominatorial parameter
_↩ _x	The argument

hypergf()

```
float __gnu_cxx::hypergf (
             float __a,
             float __b,
             float __c,
             float __x)  [inline]
```

Return the hypergeometric function ${}_2F_1(a, b; c; x)$ of @ float numeratorial parameters a and b, denominatorial parameter c, and argument x.

See also

hyperg for details.

hypergl()

```
long double __gnu_cxx::hypergl (
             long double __a,
             long double __b,
             long double __c,
             long double __x)  [inline]
```

Return the hypergeometric function ${}_2F_1(a, b; c; x)$ of long double numeratorial parameters a and b, denominatorial parameter c, and argument x.

See also

hyperg for details.

laguerre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::laguerre (
             unsigned int __n,
             _Tp __x)  [inline]
```

Returns the Laguerre polynomial $L_n(x)$ of nonnegative degree n and real argument $x >= 0$.
The Laguerre polynomial is defined by:

$$L_n(x) = \frac{e^x}{n!}\frac{d^n}{dx^n}(x^n e^{-x})$$

Template Parameters

_Tp	The floating-point type of the argument __x.

Parameters

_n	The nonnegative order
_x	The argument __x >= 0

Exceptions

std::domain_error	if __x < 0 .

laguerref()

```
float std::laguerref (
            unsigned int __n,
            float __x)  [inline]
```

Returns the Laguerre polynomial $L_n(x)$ of nonnegative degree n and float argument $x >= 0$.

See also

laguerre for more details.

laguerrel()

```
long double std::laguerrel (
            unsigned int __n,
            long double __x)  [inline]
```

Returns the Laguerre polynomial $L_n(x)$ of nonnegative degree n and long double argument $x >= 0$.

See also

laguerre for more details.

legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::legendre (
            unsigned int __l,
            _Tp __x)  [inline]
```

Return the Legendre polynomial $P_l(x)$ of nonnegative degree l and real argument $|x| <= 0$.
The Legendre function of order l and argument x, $P_l(x)$, is defined by:

$$P_l(x) = \frac{1}{2^l l!}\frac{d^l}{dx^l}(x^2 - 1)^l$$

Template Parameters

_Tp	The floating-point type of the argument __x.

Parameters

_↩ _l	The degree $l >= 0$

_↩ _x	The argument abs(__x) <= 1

Exceptions

std::domain_error	if abs(__x) > 1

legendref()

```
float std::legendref (
            unsigned int __l,
            float __x)  [inline]
```

Return the Legendre polynomial $P_l(x)$ of nonnegative degree l and float argument $|x| <= 0$.

See also

legendre for more details.

legendrel()

```
long double std::legendrel (
            unsigned int __l,
            long double __x)  [inline]
```

Return the Legendre polynomial $P_l(x)$ of nonnegative degree l and long double argument $|x| <= 0$.

See also

legendre for more details.

riemann_zeta()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::riemann_zeta (
            _Tp __s)  [inline]
```

Return the Riemann zeta function $\zeta(s)$ for real argument s.
The Riemann zeta function is defined by:

$$\zeta(s) = \sum_{k=1}^{\infty} k^{-s} \text{ for } s > 1$$

and

$$\zeta(s) = \frac{1}{1-2^{1-s}} \sum_{k=1}^{\infty} (-1)^{k-1} k^{-s} \text{ for } 0 <= s <= 1$$

For s < 1 use the reflection formula:

$$\zeta(s) = 2^s \pi^{s-1} \sin(\frac{\pi s}{2}) \Gamma(1-s) \zeta(1-s)$$

Template Parameters

_Tp	The floating-point type of the argument __s.

Parameters

_↩ _s	The argument s != 1

riemann_zetaf()

```
float std::riemann_zetaf (
            float __s)  [inline]
```

Return the Riemann zeta function $\zeta(s)$ for float argument s.

See also

riemann_zeta for more details.

riemann_zetal()

```
long double std::riemann_zetal (
            long double __s)  [inline]
```

Return the Riemann zeta function $\zeta(s)$ for long double argument s.

See also

riemann_zeta for more details.

sph_bessel()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::sph_bessel (
            unsigned int __n,
            _Tp __x)  [inline]
```

Return the spherical Bessel function $j_n(x)$ of nonnegative order n and real argument $x >= 0$.
The spherical Bessel function is defined by:

$$j_n(x) = \left(\frac{\pi}{2x}\right)^{1/2} J_{n+1/2}(x)$$

Template Parameters

_Tp	The floating-point type of the argument __x.

Parameters

_↩ _n	The integral order n >= 0
_↩ _x	The real argument x >= 0

Exceptions

std::domain_error	if __x < 0 .

sph_besself()

```
float std::sph_besself (
            unsigned int __n,
            float __x)  [inline]
```

Return the spherical Bessel function $j_n(x)$ of nonnegative order n and `float` argument $x >= 0$.

See also

sph_bessel for more details.

sph_bessell()

```
long double std::sph_bessell (
            unsigned int __n,
            long double __x)  [inline]
```

Return the spherical Bessel function $j_n(x)$ of nonnegative order n and `long double` argument $x >= 0$.

See also

sph_bessel for more details.

sph_legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::sph_legendre (
            unsigned int __l,
            unsigned int __m,
            _Tp __theta)  [inline]
```

Return the spherical Legendre function of nonnegative integral degree `l` and order `m` and real angle θ in radians. The spherical Legendre function is defined by

$$Y_l^m(\theta,\phi) = (-1)^m[\frac{(2l+1)}{4\pi}\frac{(l-m)!}{(l+m)!}]P_l^m(\cos\theta)\exp^{im\phi}$$

Template Parameters

_Tp	The floating-point type of the angle __theta.

Parameters

__l	The order __l >= 0

__m	The degree __m >= 0 and __m <= __l
__theta	The radian polar angle argument

sph_legendref()

```
float std::sph_legendref (
            unsigned int __l,
            unsigned int __m,
            float __theta)  [inline]
```

Return the spherical Legendre function of nonnegative integral degree l and order m and float angle θ in radians.

See also

sph_legendre for details.

sph_legendrel()

```
long double std::sph_legendrel (
            unsigned int __l,
            unsigned int __m,
            long double __theta)  [inline]
```

Return the spherical Legendre function of nonnegative integral degree l and order m and long double angle θ in radians.

See also

sph_legendre for details.

sph_neumann()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::sph_neumann (
            unsigned int __n,
            _Tp __x)  [inline]
```

Return the spherical Neumann function of integral order $n >= 0$ and real argument $x >= 0$.
The spherical Neumann function is defined by

$$n_n(x) = \left(\frac{\pi}{2x}\right)^{1/2} N_{n+1/2}(x)$$

Template Parameters

_Tp	The floating-point type of the argument __x.

Parameters

_↩ _n	The integral order n >= 0

_↩ _x	The real argument __x >= 0

Exceptions

std::domain_error	if __x < 0 .

sph_neumannf()

```
float std::sph_neumannf (
            unsigned int __n,
            float __x)  [inline]
```

Return the spherical Neumann function of integral order $n >= 0$ and `float` argument $x >= 0$.

See also

sph_neumann for details.

sph_neumannl()

```
long double std::sph_neumannl (
            unsigned int __n,
            long double __x)  [inline]
```

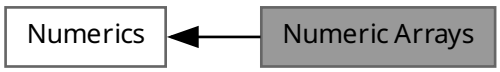
Return the spherical Neumann function of integral order $n >= 0$ and `long double` $x >= 0$.

See also

sph_neumann for details.

3.11.6 Numeric Arrays

Collaboration diagram for Numeric Arrays:



Classes

- class std::gslice
- class std::gslice_array< _Tp >
- class std::indirect_array< _Tp >
- class std::mask_array< _Tp >
- class std::slice
- class std::slice_array< _Tp >
- class std::valarray< _Tp >

Functions

- std::gslice::gslice ()
- std::gslice::gslice (const gslice &)
- std::gslice::gslice (size_t __o, const valarray< size_t > &__l, const valarray< size_t > &__s)
- std::gslice_array< _Tp >::gslice_array (const gslice_array &)
- std::indirect_array< _Tp >::indirect_array (const indirect_array &)
- std::mask_array< _Tp >::mask_array (const mask_array &)
- std::slice::slice ()
- std::slice::slice (size_t __o, size_t __d, size_t __s)
- std::slice_array< _Tp >::slice_array (const slice_array &)
- std::valarray< _Tp >::valarray () noexcept
- template<class _Dom>
  std::valarray< _Tp >::valarray (const _Expr< _Dom, _Tp > &__e)
- std::valarray< _Tp >::valarray (const _Tp &, size_t)
- template<typename _Tp>
  std::valarray< _Tp >::valarray (const _Tp *__restrict__ __p, size_t __n)
- std::valarray< _Tp >::valarray (const gslice_array< _Tp > &)
- std::valarray< _Tp >::valarray (const indirect_array< _Tp > &)
- std::valarray< _Tp >::valarray (const mask_array< _Tp > &)
- std::valarray< _Tp >::valarray (const slice_array< _Tp > &)
- std::valarray< _Tp >::valarray (const valarray &)
- std::valarray< _Tp >::valarray (initializer_list< _Tp >)
- std::valarray< _Tp >::valarray (size_t)
- std::valarray< _Tp >::valarray (valarray &&) noexcept
- std::gslice::~gslice ()
- _Expr< _ValFunClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (_Tp __func(_Tp)) const
- _Expr< _RefFunClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (_Tp __func(const _Tp &)) const
- valarray< _Tp > std::valarray< _Tp >::cshift (int __n) const
- _Tp std::valarray< _Tp >::max () const
- _Tp std::valarray< _Tp >::min () const
- gslice & std::gslice::operator= (const gslice &)
- template<class _Dom>
  void **std::gslice_array< _Tp >::operator=** (const _Expr< _Dom, _Tp > &) const
- void std::gslice_array< _Tp >::operator= (const _Tp &) const
- gslice_array & std::gslice_array< _Tp >::operator= (const gslice_array &)
- void std::gslice_array< _Tp >::operator= (const valarray< _Tp > &) const
- template<class _Dom>
  void **std::indirect_array< _Tp >::operator=** (const _Expr< _Dom, _Tp > &) const
- void std::indirect_array< _Tp >::operator= (const _Tp &) const
- indirect_array & std::indirect_array< _Tp >::operator= (const indirect_array &)
- void std::indirect_array< _Tp >::operator= (const valarray< _Tp > &) const
- template<class _Ex>
  void **std::mask_array< _Tp >::operator=** (const _Expr< _Ex, _Tp > &__e) const
- void std::mask_array< _Tp >::operator= (const _Tp &) const
- mask_array & std::mask_array< _Tp >::operator= (const mask_array &)
- void **std::mask_array< _Tp >::operator=** (const valarray< _Tp > &) const
- template<class _Dom>
  void **std::slice_array< _Tp >::operator=** (const _Expr< _Dom, _Tp > &) const
- void std::slice_array< _Tp >::operator= (const _Tp &) const
- slice_array & std::slice_array< _Tp >::operator= (const slice_array &)
- void std::slice_array< _Tp >::operator= (const valarray< _Tp > &) const

- template<class _Dom>
  valarray< _Tp > & **std::valarray< _Tp >::operator=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const _Tp &__t)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const gslice_array< _Tp > &__ga)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const indirect_array< _Tp > &__ia)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const mask_array< _Tp > &__ma)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const slice_array< _Tp > &__sa)
- valarray< _Tp > & std::valarray< _Tp >::operator= (const valarray< _Tp > &__v)
- valarray & std::valarray< _Tp >::operator= (initializer_list< _Tp > __l)
- valarray< _Tp > & std::valarray< _Tp >::operator= (valarray< _Tp > &&__v) noexcept
- gslice_array< _Tp > std::valarray< _Tp >::operator[] (const gslice &__s)
- _Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (const gslice &__s) const
- mask_array< _Tp > std::valarray< _Tp >::operator[] (const valarray< bool > &__m)
- valarray< _Tp > std::valarray< _Tp >::operator[] (const valarray< bool > &__m) const
- indirect_array< _Tp > std::valarray< _Tp >::operator[] (const valarray< size_t > &__i)
- _Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (const valarray< size_t > &__i) const
- _Tp & std::valarray< _Tp >::operator[] (size_t __i) noexcept
- const _Tp & **std::valarray< _Tp >::operator[]** (size_t) const noexcept
- slice_array< _Tp > std::valarray< _Tp >::operator[] (slice __s)
- _Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (slice __s) const
- void std::valarray< _Tp >::resize (size_t __size, _Tp __c=_Tp())
- valarray< _Tp > std::valarray< _Tp >::shift (int __n) const
- valarray< size_t > std::gslice::size () const
- size_t std::slice::size () const
- size_t std::valarray< _Tp >::size () const
- size_t std::gslice::start () const
- size_t std::slice::start () const
- valarray< size_t > std::gslice::stride () const
- size_t std::slice::stride () const
- _Tp std::valarray< _Tp >::sum () const
- void std::valarray< _Tp >::swap (valarray< _Tp > &__v) noexcept
- template<typename _Tp, size_t _Nm>
  std::valarray (const _Tp(&)[_Nm], size_t) -> valarray< _Tp >

3.11.6.1 Detailed Description

Classes and functions for representing and manipulating arrays of elements.

3.11.6.2 Function Documentation

gslice() [1/3]

```
std::gslice::gslice ()  [inline]
```

Construct an empty slice.
References gslice().
Referenced by gslice(), gslice(), gslice(), and operator=().

gslice() [2/3]

```
std::gslice::gslice (
            const gslice & __g)  [inline]
```

Copy constructor.
References gslice().

gslice() [3/3]

```
std::gslice::gslice (
            size_t __o,
            const valarray< size_t > & __l,
            const valarray< size_t > & __s)  [inline]
```

Construct a slice.
Constructs a slice with as many dimensions as the length of the *l* and *s* arrays.

Parameters

_↩ _*o*	Offset in array of first element.
_↩ _*l*	Array of dimension lengths.
_↩ _*s*	Array of dimension strides between array elements.

References gslice().

gslice_array()

```
template<typename _Tp>
std::gslice_array< _Tp >::gslice_array (
            const gslice_array< _Tp > & __a)  [inline]
```

Copy constructor. Both slices refer to the same underlying array.
References gslice_array().
Referenced by gslice_array(), and operator=().

indirect_array()

```
template<typename _Tp>
std::indirect_array< _Tp >::indirect_array (
            const indirect_array< _Tp > & __a)  [inline]
```

Copy constructor. Both slices refer to the same underlying array.
References indirect_array().
Referenced by indirect_array(), and operator=().

mask_array()

```
template<typename _Tp>
std::mask_array< _Tp >::mask_array (
            const mask_array< _Tp > & __a)  [inline]
```

Copy constructor. Both slices refer to the same underlying array.
References mask_array().
Referenced by mask_array(), and operator=().

slice() [1/2]

```
std::slice::slice ()  [inline]
```

Construct an empty slice.

Referenced by operator==.

slice() [2/2]

```
std::slice::slice (
            size_t __o,
            size_t __d,
            size_t __s)  [inline]
```

Construct a slice.

Parameters

_↩ _o	Offset in array of first element.
_↩ _d	Number of elements in slice.
_↩ _s	Stride between array elements.

slice_array()

```
template<typename _Tp>
std::slice_array< _Tp >::slice_array (
            const slice_array< _Tp > & __a)  [inline]
```

Copy constructor. Both slices refer to the same underlying array.
References slice_array().
Referenced by slice_array(), and operator=().

valarray() [1/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray ()  [inline], [noexcept]
```

Construct an empty array.
Referenced by valarray(), valarray(), valarray(), cshift(), operator%=(), operator%=(), operator&=(), operator&=(), operator*=(), operator*=(), operator+=(), operator+=(), operator-=(), operator-=(), operator/=(), operator/=(), operator<<=(), operator=(), operator=(), operator>>=(), operator>>=(), operator[](), operator[](), operator[]() const, operator[]() const, operator^=(), operator^=(), operator|=(), shift(), and swap().

valarray() [2/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            const _Tp & __t,
            size_t __n)  [inline]
```

Construct an array with *n* elements initialized to *t*.

valarray() [3/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            const gslice_array< _Tp > & __ga)  [inline]
```

Construct an array with the same size and values in *ga*.
References size().

valarray() [4/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            const indirect_array< _Tp > & __ia)  [inline]
```

Construct an array with the same size and values in *ia*.

valarray() [5/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            const mask_array< _Tp > & __ma)  [inline]
```

Construct an array with the same size and values in *ma*.

valarray() [6/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            const slice_array< _Tp > & __sa)  [inline]
```

Construct an array with the same size and values in *sa*.

valarray() [7/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            const valarray< _Tp > & __v)  [inline]
```

Copy constructor.
References valarray().

valarray() [8/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            initializer_list< _Tp > __l)  [inline]
```

Construct an array with an initializer_list of values.
References size().

valarray() [9/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            size_t __n)  [inline], [explicit]
```

Construct an array with *n* elements.

valarray() [10/10]

```
template<typename _Tp>
std::valarray< _Tp >::valarray (
            valarray< _Tp > && __v)  [inline], [noexcept]
```

Move constructor.
References valarray().

~gslice()

```
std::gslice::~gslice ()  [inline]
```

Destructor.

apply() [1/2]

```
template<class _Tp>
_Expr< _ValFunClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (
            _Tp  __func_Tp) const  [inline]
```

Apply a function to the array.
Returns a new valarray with elements assigned to the result of applying __func to the corresponding element of this array. The new array has the same size as this one.

Parameters

__func	Function of Tp returning Tp to apply.

Returns

New valarray with transformed elements.

apply() [2/2]

```
template<class _Tp>
_Expr< _RefFunClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::apply (
            _Tp  __funcconst _Tp &) const  [inline]
```

Apply a function to the array.
Returns a new valarray with elements assigned to the result of applying __func to the corresponding element of this array. The new array has the same size as this one.

Parameters

__func	Function of const Tp& returning Tp to apply.

Returns

New valarray with transformed elements.

cshift()

```
template<class _Tp>
valarray< _Tp > std::valarray< _Tp >::cshift (
            int __n) const  [inline]
```

Return a rotated array.
A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index i, the new position is (i - n) % size(). The new valarray has the same size as the current one. Elements that are shifted beyond the array bounds are shifted into the other end of the array. No elements are lost.
Positive arguments shift toward index 0, wrapping around the top. Negative arguments shift towards the top, wrapping around to 0.

Parameters

_↩ _n	Number of element positions to rotate.

Returns

New valarray with elements in shifted positions.

References valarray().

max()

```
template<typename _Tp>
_Tp std::valarray< _Tp >::max () const  [inline]
```

Return the maximum element using operator<().

min()

```
template<typename _Tp>
_Tp std::valarray< _Tp >::min () const  [inline]
```

Return the minimum element using operator<().

operator=() [1/20]

```
gslice & std::gslice::operator= (
            const gslice & __g)  [inline]
```

Assignment operator.
References gslice().

operator=() [2/20]

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator= (
            const _Tp & __t) const  [inline]
```

Assign all slice elements to *t*.

operator=() [3/20]

```
template<typename _Tp>
gslice_array< _Tp > & std::gslice_array< _Tp >::operator= (
            const gslice_array< _Tp > & __a)  [inline]
```

Assignment operator. Assigns slice elements to corresponding elements of *a*.
References gslice_array().

operator=() [4/20]

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator= (
            const valarray< _Tp > & __v) const  [inline]
```

Assign slice elements to corresponding elements of *v*.
References std::valarray< _Tp >::size().

operator=() [5/20]

```
template<typename _Tp>
void std::indirect_array< _Tp >::operator= (
            const _Tp & __t) const  [inline]
```

Assign all slice elements to *t*.

operator=() [6/20]

```
template<typename _Tp>
indirect_array< _Tp > & std::indirect_array< _Tp >::operator= (
            const indirect_array< _Tp > & __a)  [inline]
```

Assignment operator. Assigns elements to corresponding elements of *a*.
References indirect_array().

operator=() [7/20]

```
template<typename _Tp>
void std::indirect_array< _Tp >::operator= (
            const valarray< _Tp > & __v) const  [inline]
```

Assign slice elements to corresponding elements of *v*.

operator=() [8/20]

```
template<typename _Tp>
void std::mask_array< _Tp >::operator= (
            const _Tp & __t) const  [inline]
```

Assign all slice elements to *t*.

operator=() [9/20]

```
template<typename _Tp>
mask_array< _Tp > & std::mask_array< _Tp >::operator= (
            const mask_array< _Tp > & __a)  [inline]
```

Assignment operator. Assigns elements to corresponding elements of *a*.
References mask_array().

operator=() [10/20]

```
template<typename _Tp>
void std::slice_array< _Tp >::operator= (
            const _Tp & __t) const  [inline]
```

Assign all slice elements to *t*.

operator=() [11/20]

```
template<typename _Tp>
slice_array< _Tp > & std::slice_array< _Tp >::operator= (
            const slice_array< _Tp > & __a)  [inline]
```

Assignment operator. Assigns slice elements to corresponding elements of *a*.
References slice_array().

operator=() [12/20]

```
template<typename _Tp>
void std::slice_array< _Tp >::operator= (
            const valarray< _Tp > & __v) const  [inline]
```

Assign slice elements to corresponding elements of *v*.

operator=() [13/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            const _Tp & __t)  [inline]
```

Assign elements to a value.
Assign all elements of array to *t*.

Parameters

↩ _↩ ↩ _↩ *t*	Value for elements.

operator=() [14/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            const gslice_array< _Tp > & __ga)  [inline]
```

Assign elements to an array subset.
Assign elements of array to values in *ga*. Results are undefined if *ga* does not have the same size as this array.

Parameters

__ga	Array slice to get values from.

References size().

operator=() [15/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            const indirect_array< _Tp > & __ia)  [inline]
```

Assign elements to an array subset.
Assign elements of array to values in *ia*. Results are undefined if *ia* does not have the same size as this array.

Parameters

_↩ *_ia*	Array slice to get values from.

operator=() [16/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            const mask_array< _Tp > & __ma)  [inline]
```

Assign elements to an array subset.
Assign elements of array to values in *ma*. Results are undefined if *ma* does not have the same size as this array.

Parameters

__ma	Array slice to get values from.

operator=() [17/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            const slice_array< _Tp > & __sa)  [inline]
```

Assign elements to an array subset.
Assign elements of array to values in *sa*. Results are undefined if *sa* does not have the same size as this array.

Parameters

__sa	Array slice to get values from.

operator=() [18/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            const valarray< _Tp > & __v)  [inline]
```

Assign elements to an array.
Assign elements of array to values in *v*.

Parameters

_↩ _v	Valarray to get values from.

References valarray().

operator=() [19/20]

```
template<typename _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator= (
            initializer_list< _Tp > __l)  [inline]
```

Assign elements to an initializer_list.
Assign elements of array to values in *__l*. Results are undefined if *__l* does not have the same size as this array.

Parameters

↩ _↩ ↩ _↩ l	initializer_list to get values from.

operator=() [20/20]

```
template<typename _Tp>
```

```
valarray< _Tp > & std::valarray< _Tp >::operator= (
            valarray< _Tp > && __v)  [inline], [noexcept]
```

Move assign elements to an array.
Move assign elements of array to values in *v*.

Parameters

_↩ _v	Valarray to get values from.

References valarray().

operator[]() [1/9]

```
template<typename _Tp>
gslice_array< _Tp > std::valarray< _Tp >::operator[] (
            const gslice & __s)  [inline]
```

Return a reference to an array subset.
Returns a new valarray containing the elements of the array indicated by the gslice argument. The new valarray has the same size as the input gslice.

See also

gslice.

Parameters

_↩ _s	The source gslice.

Returns

New valarray containing elements in __s.

operator[]() [2/9]

```
template<typename _Tp>
_Expr< _GClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
            const gslice & __s) const  [inline]
```

Return an array subset.
Returns a slice_array referencing the elements of the array indicated by the slice argument.

See also

gslice.

Parameters

_↩ _s	The source slice.

Returns

Slice_array referencing elements indicated by __*s*.

operator[]() [3/9]

```
template<typename _Tp>
mask_array< _Tp > std::valarray< _Tp >::operator[] (
            const valarray< bool > & __m)  [inline]
```

Return a reference to an array subset.
Returns a new mask_array referencing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements are part of the subset. Elements of the array are part of the subset if the corresponding element of the argument is true.

Parameters

_↩ _*m*	The valarray bitmask.

Returns

New valarray containing elements indicated by __*m*.

References valarray(), and size().

operator[]() [4/9]

```
template<typename _Tp>
valarray< _Tp > std::valarray< _Tp >::operator[] (
            const valarray< bool > & __m) const  [inline]
```

Return an array subset.
Returns a new valarray containing the elements of the array indicated by the argument. The input is a valarray of bool which represents a bitmask indicating which elements should be copied into the new valarray. Each element of the array is added to the return valarray if the corresponding element of the argument is true.

Parameters

_↩ _*m*	The valarray bitmask.

Returns

New valarray containing elements indicated by __*m*.

References valarray(), and size().

operator[]() [5/9]

```
template<typename _Tp>
indirect_array< _Tp > std::valarray< _Tp >::operator[] (
            const valarray< size_t > & __i)  [inline]
```

Return a reference to an array subset.
Returns an indirect_array referencing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to include in the subset. The returned indirect_array refers to these elements.

Parameters

↩ _↩ ↩ _↩ *i*	The valarray element index list.

Returns

Indirect_array referencing elements in *__i*.

References valarray(), and size().

operator[]() [6/9]

```
template<typename _Tp>
_Expr< _IClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
            const valarray< size_t > & __i) const  [inline]
```

Return an array subset.
Returns a new valarray containing the elements of the array indicated by the argument. The elements in the argument are interpreted as the indices of elements of this valarray to copy to the return valarray.

Parameters

↩ _↩ ↩ _↩ *i*	The valarray element index list.

Returns

New valarray containing elements in *__s*.

References valarray().

operator[]() [7/9]

```
template<typename _Tp>
_Tp & std::valarray< _Tp >::operator[] (
            size_t __i)  [inline], [noexcept]
```

Return a reference to the i'th array element.

Parameters

↩ _↩ ↩ _↩ *i*	Index of element to return.

Returns

Reference to the i'th element.

operator[]() [8/9]

```
template<typename _Tp>
slice_array< _Tp > std::valarray< _Tp >::operator[] (
            slice __s)  [inline]
```

Return a reference to an array subset.
Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

See also

slice.

Parameters

_↩ _s	The source slice.

Returns

New valarray containing elements in __s.

operator[]() [9/9]

```
template<typename _Tp>
_Expr< _SClos< _ValArray, _Tp >, _Tp > std::valarray< _Tp >::operator[] (
            slice __s) const  [inline]
```

Return an array subset.
Returns a new valarray containing the elements of the array indicated by the slice argument. The new valarray has the same size as the input slice.

See also

slice.

Parameters

_↩ _s	The source slice.

Returns

New valarray containing elements in __s.

resize()

```
template<class _Tp>
void std::valarray< _Tp >::resize (
```

```
            size_t __size,
            _Tp __c = _Tp())  [inline]
```

Resize array.

Resize this array to *size* and set all elements to *c*. All references and iterators are invalidated.

Parameters

__size	New array size.
__c	New value for all elements.

shift()

```
template<class _Tp>
valarray< _Tp > std::valarray< _Tp >::shift (
            int __n) const  [inline]
```

Return a shifted array.

A new valarray is constructed as a copy of this array with elements in shifted positions. For an element with index i, the new position is i - n. The new valarray has the same size as the current one. New elements without a value are set to 0. Elements whose new position is outside the bounds of the array are discarded.

Positive arguments shift toward index 0, discarding elements [0, n). Negative arguments discard elements from the top of the array.

Parameters

_↩ _n	Number of element positions to shift.

Returns

New valarray with elements in shifted positions.

References valarray().

size() [1/3]

```
valarray< size_t > std::gslice::size () const  [inline]
```

Return array of sizes of slice dimensions.

size() [2/3]

```
size_t std::slice::size () const  [inline]
```

Return size of slice.

size() [3/3]

```
template<class _Tp>
size_t std::valarray< _Tp >::size () const  [inline]
```

Return the number of elements in array.

Referenced by valarray(), valarray(), std::gslice_array< _Tp >::operator=(), operator=(), operator[](), operator[](), and operator[]().

start() [1/2]

```
size_t std::gslice::start () const  [inline]
```
Return array offset of first slice element.

start() [2/2]

```
size_t std::slice::start () const  [inline]
```
Return array offset of first slice element.

stride() [1/2]

```
valarray< size_t > std::gslice::stride () const  [inline]
```
Return array of array strides for each dimension.

stride() [2/2]

```
size_t std::slice::stride () const  [inline]
```
Return array stride of slice.

sum()

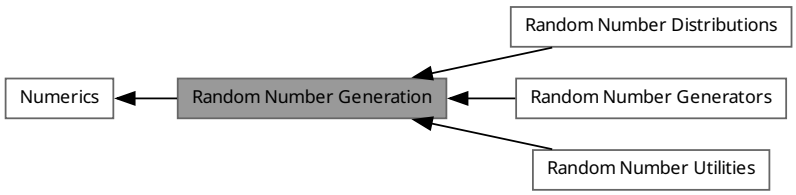
```
template<class _Tp>
_Tp std::valarray< _Tp >::sum () const  [inline]
```
Return the sum of all elements in the array.
Accumulates the sum of all elements into a Tp using +=. The order of adding the elements is unspecified.

swap()

```
template<class _Tp>
void std::valarray< _Tp >::swap (
            valarray< _Tp > & __v)  [inline], [noexcept]
```
Swap.
References valarray().

3.11.7 Random Number Generation

Collaboration diagram for Random Number Generation:



Topics

- Random Number Distributions
- Random Number Generators
- Random Number Utilities

Functions

- template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>
  _RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)

3.11.7.1 Detailed Description

A facility for generating random numbers on selected distributions.

3.11.7.2 Function Documentation

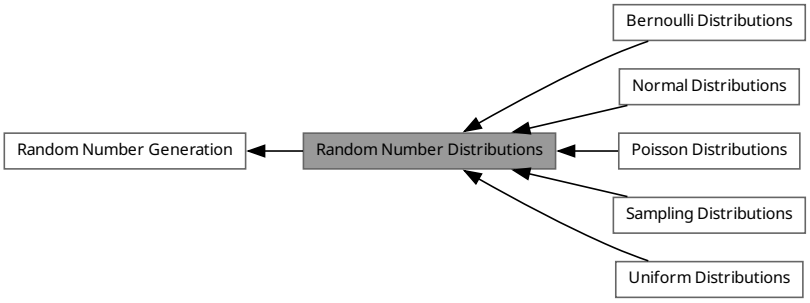
generate_canonical()

```
template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>
_RealType std::generate_canonical (
            _UniformRandomNumberGenerator & __g)
```

A function template for converting the output of a (integral) uniform random number generator to a floatng point result in the range [0-1).
Referenced by generate_canonical().

3.11.7.3 Random Number Distributions

Collaboration diagram for Random Number Distributions:



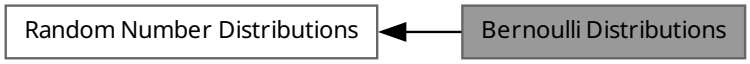
Topics

- Bernoulli Distributions
- Normal Distributions
- Poisson Distributions
- Sampling Distributions
- Uniform Distributions

3.11.7.3.1 Detailed Description

3.11.7.3.2 Bernoulli Distributions

Collaboration diagram for Bernoulli Distributions:



Classes

- class std::bernoulli_distribution
- class std::binomial_distribution< _IntType >
- class std::geometric_distribution< _IntType >
- class std::negative_binomial_distribution< _IntType >

Functions

- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)
- template<typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::bernoulli_distribution &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)

Detailed Description

Function Documentation

operator<<() [1/2]

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::bernoulli_distribution & __x)
```

Inserts a bernoulli_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A bernoulli_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator<<() [2/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::geometric_distribution< _IntType > & __x)
```

Inserts a geometric_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A geometric_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

References std::numeric_limits< _Tp >::epsilon(), log(), and std::numeric_limits< _Tp >::max().

operator>>() [1/2]

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::bernoulli_distribution & __x)  [inline]
```

Extracts a bernoulli_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A bernoulli_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

References std::bernoulli_distribution::param().

operator>>() [2/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::geometric_distribution< _IntType > & __x)
```

Extracts a geometric_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _*is*	An input stream.
_↩ _*x*	A geometric_distribution random number generator engine.

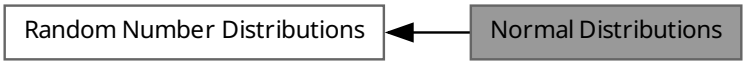
Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::geometric_distribution< _IntType >::param().

3.11.7.3.3 Normal Distributions

Collaboration diagram for Normal Distributions:



Classes

- class std::cauchy_distribution< _RealType >
- class std::chi_squared_distribution< _RealType >
- class std::fisher_f_distribution< _RealType >
- class std::lognormal_distribution< _RealType >
- class std::normal_distribution< _RealType >
- class std::student_t_distribution< _RealType >

Functions

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)

Detailed Description

Function Documentation

operator<<()

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::cauchy_distribution< _RealType > & __x)
```

Inserts a cauchy_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A cauchy_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

References tan().

operator>>()

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::cauchy_distribution< _RealType > & __x)
```

Extracts a cauchy_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A cauchy_distribution random number generator engine.

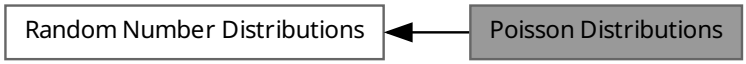
Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::cauchy_distribution< _RealType >::param().

3.11.7.3.4 Poisson Distributions

Collaboration diagram for Poisson Distributions:



Classes

- class std::exponential_distribution< _RealType >
- class std::extreme_value_distribution< _RealType >
- class std::gamma_distribution< _RealType >
- class std::poisson_distribution< _IntType >
- class std::weibull_distribution< _RealType >

Functions

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)

Detailed Description

Function Documentation

operator<<() [1/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::exponential_distribution< _RealType > & __x)
```

Inserts a exponential_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A exponential_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

References log().

operator<<() [2/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::extreme_value_distribution< _RealType > & __x)
```

Inserts a extreme_value_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A extreme_value_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

References log().

operator<<() [3/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::weibull_distribution< _RealType > & __x)
```

Inserts a weibull_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A weibull_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

References log(), and pow().

operator>>() [1/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::exponential_distribution< _RealType > & __x)
```

Extracts a exponential_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A exponential_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::exponential_distribution< _RealType >::param().

operator>>() [2/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::extreme_value_distribution< _RealType > & __x)
```

Extracts a extreme_value_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A extreme_value_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::extreme_value_distribution< _RealType >::param().

operator>>() [3/3]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::weibull_distribution< _RealType > & __x)
```

Extracts a weibull_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A weibull_distribution random number generator engine.

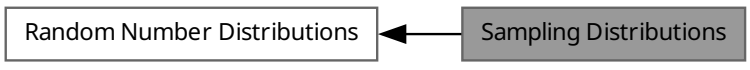
Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::weibull_distribution< _RealType >::param().

3.11.7.3.5 Sampling Distributions

Collaboration diagram for Sampling Distributions:



Namespaces

- namespace std::__detail

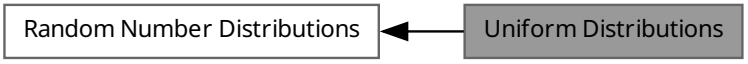
Classes

- class std::discrete_distribution< _IntType >
- class std::piecewise_constant_distribution< _RealType >
- class std::piecewise_linear_distribution< _RealType >

Detailed Description

3.11.7.3.6 Uniform Distributions

Collaboration diagram for Uniform Distributions:



Concepts

- concept std::uniform_random_bit_generator

Classes

- class std::uniform_int_distribution< _IntType >
- class std::uniform_real_distribution< _RealType >

Functions

- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)

Detailed Description

Function Documentation

operator<<() [1/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
```

```
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::uniform_int_distribution< _IntType > & __x)
```

Inserts a uniform_int_distribution random number distribution __x into the output stream os.

Parameters

__os	An output stream.
__x	A uniform_int_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator<<() [2/2]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::uniform_real_distribution< _RealType > & __x)
```

Inserts a uniform_real_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A uniform_real_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator>>() [1/2]

```
template<typename _IntType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::uniform_int_distribution< _IntType > & __x)
```

Extracts a uniform_int_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A uniform_int_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::uniform_int_distribution< _IntType >::param().

operator>>() [2/2]

```
template<typename _RealType, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::uniform_real_distribution< _RealType > & __x)
```

Extracts a uniform_real_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A uniform_real_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

References std::ios_base::flags(), and std::uniform_real_distribution< _RealType >::param().

3.11.7.4 Random Number Generators

Collaboration diagram for Random Number Generators:



Classes

- class std::discard_block_engine< _RandomNumberEngine, __p, __r >
- class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >
- class std::linear_congruential_engine< _UIntType, __a, __c, __m >
- class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >
- class std::philox_engine< _UIntType, __w, __n, __r, __consts >
- class std::random_device
- class std::shuffle_order_engine< _RandomNumberEngine, __k >
- class std::subtract_with_carry_engine< _UIntType, __w, __s, __r >

Typedefs

- typedef minstd_rand0 **std::default_random_engine**
- typedef shuffle_order_engine< minstd_rand0, 256 > **std::knuth_b**
- typedef linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > std::minstd_rand
- typedef linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > std::minstd_rand0
- typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > std::mt19937
- typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67fffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > std::mt19937_64
- typedef philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 > std::philox4x32
- typedef philox_engine< uint_fast64_t, 64, 4, 10, 0xCA5A826395121157, 0x9E3779B97F4A7C15, 0xD2E7470↩ EE14C6C93, 0xBB67AE8584CAA73B > std::philox4x64
- typedef discard_block_engine< ranlux24_base, 223, 23 > **std::ranlux24**
- typedef subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 > **std::ranlux24_base**
- typedef discard_block_engine< ranlux48_base, 389, 11 > **std::ranlux48**
- typedef subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 > **std::ranlux48_base**

Functions

- template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits> std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)

3.11.7.4.1 Detailed Description

These classes define objects which provide random or pseudorandom numbers, either from a discrete or a continuous interval. The random number generator supplied as a part of this library are all uniform random number generators which provide a sequence of random number uniformly distributed over their range.
A number generator is a function object with an operator() that takes zero arguments and returns a number.
A compliant random number generator must satisfy the following requirements.

Table 287: Random Number Generator Requirements

To be documented.

3.11.7.4.2 Typedef Documentation

minstd_rand

```
typedef linear_congruential_engine<uint_fast32_t, 48271UL, 0UL, 2147483647UL> std::minstd_rand
```

An alternative LCR (Lehmer Generator function).

minstd_rand0

```
typedef linear_congruential_engine<uint_fast32_t, 16807UL, 0UL, 2147483647UL> std::minstd_rand0
```

The classic Minimum Standard rand0 of Lewis, Goodman, and Miller.

mt19937

```
typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL,
7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL> std::mt19937
```

The classic Mersenne Twister.
Reference: M. Matsumoto and T. Nishimura, Mersenne Twister: A 623-Dimensionally Equidistributed Uniform Pseudo-Random Number Generator, ACM Transactions on Modeling and Computer Simulation, Vol. 8, No. 1, January 1998, pp 3-30.

mt19937_64

```
typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29,
0x5555555555555555ULL, 17, 0x71d67fffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL>
std::mt19937_64
```

An alternative Mersenne Twister.

philox4x32

```
typedef philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85>
std::philox4x32
```

32-bit four-word Philox engine.

philox4x64

```
typedef philox_engine< uint_fast64_t, 64, 4, 10, 0xCA5A826395121157, 0x9E3779B97F4A7C15, 0xD2↩
E7470EE14C6C93, 0xBB67AE8584CAA73B> std::philox4x64
```

64-bit four-word Philox engine.

3.11.7.4.3 Function Documentation

operator<<()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename
_Traits>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x)
```

Inserts the current state of a independent_bits_engine random number generator engine __x into the output stream __os.

Parameters

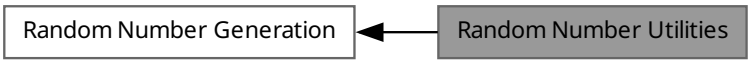
__os	An output stream.
__x	A independent_bits_engine random number generator engine.

Returns

The output stream with the state of __x inserted or in an error state.

3.11.7.5 Random Number Utilities

Collaboration diagram for Random Number Utilities:



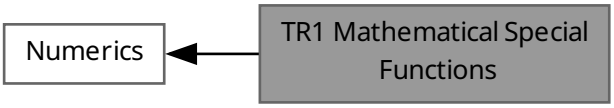
Classes

- class std::seed_seq

3.11.7.5.1 Detailed Description

3.11.8 TR1 Mathematical Special Functions

Collaboration diagram for TR1 Mathematical Special Functions:



Functions

- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_laguerref** (unsigned int __n, unsigned int __m, float __x)
- long double **std::tr1::assoc_laguerrel** (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_legendref** (unsigned int __l, unsigned int __m, float __x)
- long double **std::tr1::assoc_legendrel** (unsigned int __l, unsigned int __m, long double __x)
- template<typename _Tpx, typename _Tpy>
  __gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta (_Tpx __x, _Tpy __y)
- float **std::tr1::betaf** (float __x, float __y)
- long double **std::tr1::betal** (long double __x, long double __y)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1 (_Tp __k)

- float **std::tr1::comp_ellint_1f** (float __k)
- long double **std::tr1::comp_ellint_1l** (long double __k)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2 (_Tp __k)
- float **std::tr1::comp_ellint_2f** (float __k)
- long double **std::tr1::comp_ellint_2l** (long double __k)
- template<typename _Tp, typename _Tpn>
  __gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3 (_Tp __k, _Tpn __nu)
- float **std::tr1::comp_ellint_3f** (float __k, float __nu)
- long double **std::tr1::comp_ellint_3l** (long double __k, long double __nu)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_if** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_il** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_jf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_jl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_kf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_kl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_neumannf** (float __nu, float __x)
- long double **std::tr1::cyl_neumannl** (long double __nu, long double __x)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1 (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_1f** (float __k, float __phi)
- long double **std::tr1::ellint_1l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2 (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_2f** (float __k, float __phi)
- long double **std::tr1::ellint_2l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp>
  __gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float **std::tr1::ellint_3f** (float __k, float __nu, float __phi)
- long double **std::tr1::ellint_3l** (long double __k, long double __nu, long double __phi)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::expint (_Tp __x)
- float **std::tr1::expintf** (float __x)
- long double **std::tr1::expintl** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (unsigned int __n, _Tp __x)
- float **std::tr1::hermitef** (unsigned int __n, float __x)
- long double **std::tr1::hermitel** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (unsigned int __n, _Tp __x)
- float **std::tr1::laguerref** (unsigned int __n, float __x)
- long double **std::tr1::laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::legendre (unsigned int __n, _Tp __x)

- float **std::tr1::legendref** (unsigned int __n, float __x)
- long double **std::tr1::legendrel** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::riemann_zeta (_Tp __x)
- float **std::tr1::riemann_zetaf** (float __x)
- long double **std::tr1::riemann_zetal** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_bessel (unsigned int __n, _Tp __x)
- float **std::tr1::sph_besself** (unsigned int __n, float __x)
- long double **std::tr1::sph_bessell** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float **std::tr1::sph_legendref** (unsigned int __l, unsigned int __m, float __theta)
- long double **std::tr1::sph_legendrel** (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_neumann (unsigned int __n, _Tp __x)
- float **std::tr1::sph_neumannf** (unsigned int __n, float __x)
- long double **std::tr1::sph_neumannl** (unsigned int __n, long double __x)

3.11.8.1 Detailed Description

A collection of advanced mathematical special functions.

3.11.8.2 Function Documentation

assoc_laguerre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre (
            unsigned int __n,
            unsigned int __m,
            _Tp __x)  [inline]
```

5.2.1.1 Associated Laguerre polynomials.

assoc_legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre (
            unsigned int __l,
            unsigned int __m,
            _Tp __x)  [inline]
```

5.2.1.2 Associated Legendre functions.

beta()

```
template<typename _Tpx, typename _Tpy>
__gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta (
            _Tpx __x,
            _Tpy __y)  [inline]
```

5.2.1.3 Beta functions.

comp_ellint_1()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1 (
            _Tp __k)  [inline]
```

5.2.1.4 Complete elliptic integrals of the first kind.

comp_ellint_2()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2 (
            _Tp __k)  [inline]
```

5.2.1.5 Complete elliptic integrals of the second kind.

comp_ellint_3()

```
template<typename _Tp, typename _Tpn>
__gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3 (
            _Tp __k,
            _Tpn __nu)  [inline]
```

5.2.1.6 Complete elliptic integrals of the third kind.

cyl_bessel_i()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i (
            _Tpnu __nu,
            _Tp __x)  [inline]
```

5.2.1.8 Regular modified cylindrical Bessel functions.

cyl_bessel_j()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j (
            _Tpnu __nu,
            _Tp __x)  [inline]
```

5.2.1.9 Cylindrical Bessel functions (of the first kind).

cyl_bessel_k()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k (
            _Tpnu __nu,
            _Tp __x)  [inline]
```

5.2.1.10 Irregular modified cylindrical Bessel functions.

cyl_neumann()

```
template<typename _Tpnu, typename _Tp>
__gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann (
            _Tpnu __nu,
            _Tp __x)  [inline]
```

5.2.1.11 Cylindrical Neumann functions.

ellint_1()

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1 (
             _Tp __k,
             _Tpp __phi)  [inline]
```

5.2.1.12 Incomplete elliptic integrals of the first kind.

ellint_2()

```
template<typename _Tp, typename _Tpp>
__gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2 (
             _Tp __k,
             _Tpp __phi)  [inline]
```

5.2.1.13 Incomplete elliptic integrals of the second kind.

ellint_3()

```
template<typename _Tp, typename _Tpn, typename _Tpp>
__gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint_3 (
             _Tp __k,
             _Tpn __nu,
             _Tpp __phi)  [inline]
```

5.2.1.14 Incomplete elliptic integrals of the third kind.

expint()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::expint (
             _Tp __x)  [inline]
```

5.2.1.15 Exponential integrals.

hermite()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (
             unsigned int __n,
             _Tp __x)  [inline]
```

5.2.1.16 Hermite polynomials.

laguerre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (
             unsigned int __n,
             _Tp __x)  [inline]
```

5.2.1.18 Laguerre polynomials.

legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::legendre (
             unsigned int __n,
             _Tp __x)  [inline]
```

5.2.1.19 Legendre polynomials.

riemann_zeta()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::riemann_zeta (
            _Tp __x)  [inline]
```

5.2.1.20 Riemann zeta function.

sph_bessel()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::sph_bessel (
            unsigned int __n,
            _Tp __x)  [inline]
```

5.2.1.21 Spherical Bessel functions.

sph_legendre()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::sph_legendre (
            unsigned int __l,
            unsigned int __m,
            _Tp __theta)  [inline]
```

5.2.1.22 Spherical associated Legendre functions.

sph_neumann()

```
template<typename _Tp>
__gnu_cxx::__promote< _Tp >::__type std::tr1::sph_neumann (
            unsigned int __n,
            _Tp __x)  [inline]
```

5.2.1.23 Spherical Neumann functions.

3.12 Ranges

Classes

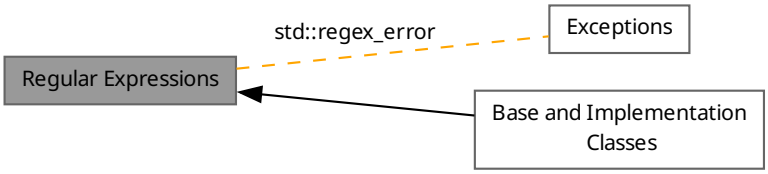
- struct std::ranges::elements_of< _Range, _Alloc >

3.12.1 Detailed Description

Components for dealing with ranges of elements.

3.13 Regular Expressions

Collaboration diagram for Regular Expressions:



Topics

- Base and Implementation Classes

Namespaces

- namespace std::regex_constants

Classes

- class std::basic_regex< _Ch_type, _Rx_traits >
- class std::match_results< _Bi_iter, _Alloc >
- class std::regex_error
- class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
- class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
- class std::regex_traits< _Ch_type >
- class std::sub_match< _BiIter >

Typedefs

- typedef match_results< const char * > **std::cmatch**
- typedef regex_iterator< const char * > **std::cregex_iterator**
- typedef regex_token_iterator< const char * > std::cregex_token_iterator
- typedef sub_match< const char * > std::csub_match
- typedef basic_regex< char > std::regex
- typedef match_results< string::const_iterator > **std::smatch**
- typedef regex_iterator< string::const_iterator > **std::sregex_iterator**
- typedef regex_token_iterator< string::const_iterator > std::sregex_token_iterator
- typedef sub_match< string::const_iterator > std::ssub_match
- typedef match_results< const wchar_t * > **std::wcmatch**
- typedef regex_iterator< const wchar_t * > **std::wcregex_iterator**
- typedef regex_token_iterator< const wchar_t * > std::wcregex_token_iterator
- typedef sub_match< const wchar_t * > std::wcsub_match
- typedef basic_regex< wchar_t > std::wregex
- typedef match_results< wstring::const_iterator > **std::wsmatch**
- typedef regex_iterator< wstring::const_iterator > **std::wsregex_iterator**
- typedef regex_token_iterator< wstring::const_iterator > std::wsregex_token_iterator
- typedef sub_match< wstring::const_iterator > std::wssub_match

Functions

- template<typename _ForwardIterator>
  std::basic_regex (_ForwardIterator, _ForwardIterator, regex_constants::syntax_option_type={}) -> basic_↩ regex< typename iterator_traits< _ForwardIterator >::value_type >
- template<typename _Bi_iter, typename _Alloc>
  bool operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &_↩ _m2)
- template<typename _Ch_type, typename _Rx_traits>
  void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs) noex- cept
- template<typename _Bi_iter, typename _Alloc>
  void swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs) noexcept

Matching, Searching, and Replacing

- template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type, typename _Alloc, typename _Rx_traits>
  bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete
- template<typename _Ch_type, class _Rx_traits>
  bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type, class _Alloc, class _Rx_traits>
  bool std::regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__↩ e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)

- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  _Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↩
  flags=regex_constants::match_default)
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
  _Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _↩
  Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa>
  basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__↩
  s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __↩
  flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx↩
  _traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↩
  flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type>
  basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)

- template<typename _BiIter>
  bool operator== (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _BiIter>
  auto operator<=> (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs) noexcept(__↩
  detail::__is_contiguous_iter< _BiIter >::value)
- template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>
  bool operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch↩
  _alloc > &__rhs)
- template<typename _Bi_iter, typename _Ch_traits, typename _Alloc>
  auto operator<=> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _↩
  Alloc > &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)
- template<typename _Bi_iter>
  bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs)
- template<typename _Bi_iter>
  auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const *__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)
- template<typename _Bi_iter>
  bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs)
- template<typename _Bi_iter>
  auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)

- template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter>
  basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const sub_match< _Bi_iter > &__m)

3.13.1 Detailed Description

A facility for performing regular expression pattern matching.

Since

C++11

3.13.2 Typedef Documentation

cregex_token_iterator

```
typedef regex_token_iterator<const char*> std::cregex_token_iterator
```

Token iterator for C-style NULL-terminated strings.

csub_match

```
typedef sub_match<const char*> std::csub_match
```

Standard regex submatch over a C-style null-terminated string.

regex

```
typedef basic_regex<char> std::regex
```

Standard regular expressions.

sregex_token_iterator

```
typedef regex_token_iterator<string::const_iterator> std::sregex_token_iterator
```

Token iterator for standard strings.

ssub_match

```
typedef sub_match<string::const_iterator> std::ssub_match
```

Standard regex submatch over a standard string.

wcregex_token_iterator

```
typedef regex_token_iterator<const wchar_t*> std::wcregex_token_iterator
```

Token iterator for C-style NULL-terminated wide strings.

wcsub_match

```
typedef sub_match<const wchar_t*> std::wcsub_match
```

Regex submatch over a C-style null-terminated wide string.

wregex

```
typedef basic_regex<wchar_t> std::wregex
```

Standard wide-character regular expressions.

wsregex_token_iterator

```
typedef regex_token_iterator<wstring::const_iterator> std::wsregex_token_iterator
```
Token iterator for standard wide-character strings.

wssub_match

```
typedef sub_match<wstring::const_iterator> std::wssub_match
```
Regex submatch over a standard wide string.

3.13.3 Function Documentation

operator<<()

```
template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter>
basic_ostream< _Ch_type, _Ch_traits > & operator<< (
            basic_ostream< _Ch_type, _Ch_traits > & __os,
            const sub_match< _Bi_iter > & __m)  [related]
```
Inserts a matched string into an output stream.

Parameters

__os	The output stream.
__m	A submatch string.

Returns

the output stream with the submatch string inserted.

operator<=>() [1/4]

```
template<typename _Bi_iter, typename _Ch_traits, typename _Alloc>
auto operator<=> (
            const sub_match< _Bi_iter > & __lhs,
            const __sub_match_string< _Bi_iter, _Ch_traits, _Alloc > & __rhs)  [related]
```
Three-way comparison of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A string.

Returns

A value indicating whether `__lhs` is less than, equal to, greater than, or incomparable with `__rhs`.

operator<=>() [2/4]

```
template<typename _Bi_iter>
auto operator<=> (
```

```
            const sub_match< _Bi_iter > & __lhs,
            typename iterator_traits< _Bi_iter >::value_type const & __rhs)  [related]
```

Three-way comparison of a regular expression submatch and a character.

Parameters

__lhs	A regular expression submatch.
__rhs	A character.

Returns

A value indicating whether __lhs is less than, equal to, greater than, or incomparable with __rhs.

operator<=>() [3/4]

```
template<typename _Bi_iter>
auto operator<=> (
            const sub_match< _Bi_iter > & __lhs,
            typename iterator_traits< _Bi_iter >::value_type const * __rhs)  [related]
```

Three-way comparison of a regular expression submatch and a C string.

Parameters

__lhs	A regular expression submatch.
__rhs	A null-terminated string.

Returns

A value indicating whether __lhs is less than, equal to, greater than, or incomparable with __rhs.

operator<=>() [4/4]

```
template<typename _BiIter>
auto operator<=> (
            const sub_match< _BiIter > & __lhs,
            const sub_match< _BiIter > & __rhs)  [related]
```

Three-way comparison of two regular expression submatches.

Parameters

__lhs	First regular expression submatch.
__rhs	Second regular expression submatch.

Returns

A value indicating whether __lhs is less than, equal to, greater than, or incomparable with __rhs.

operator==() [1/5]

```
template<typename _Bi_iter, typename _Alloc>
bool operator== (
            const match_results< _Bi_iter, _Alloc > & __m1,
            const match_results< _Bi_iter, _Alloc > & __m2)  [related]
```

Compares two match_results for equality.

Returns

true if the two objects refer to the same match, false otherwise.

operator==() [2/5]

```
template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>
bool operator== (
            const sub_match< _Bi_iter > & __lhs,
            const __sub_match_string< _Bi_iter, _Ch_traits, _Ch_alloc > & __rhs)  [related]
```

Tests the equivalence of a regular expression submatch and a string.

Parameters

__lhs	A regular expression submatch.
__rhs	A string.

Returns

true if *__lhs* is equivalent to *__rhs*, false otherwise.

operator==() [3/5]

```
template<typename _Bi_iter>
bool operator== (
            const sub_match< _Bi_iter > & __lhs,
            typename iterator_traits< _Bi_iter >::value_type const & __rhs)  [related]
```

Tests the equivalence of a regular expression submatch and a character.

Parameters

__lhs	A regular expression submatch.
__rhs	A character.

Returns

true if *__lhs* is equivalent to *__rhs*, false otherwise.

operator==() [4/5]

```
template<typename _Bi_iter>
bool operator== (
            const sub_match< _Bi_iter > & __lhs,
            typename iterator_traits< _Bi_iter >::value_type const * __rhs)  [related]
```

Tests the equivalence of a regular expression submatch and a C string.

Parameters

__lhs	A regular expression submatch.
__rhs	A null-terminated string.

Returns

true if *__lhs* is equivalent to *__rhs*, false otherwise.

operator==() [5/5]

```
template<typename _BiIter>
bool operator== (
            const sub_match< _BiIter > & __lhs,
            const sub_match< _BiIter > & __rhs)  [related]
```

Tests the equivalence of two regular expression submatches.

Parameters

__lhs	First regular expression submatch.
__rhs	Second regular expression submatch.

Returns

true if *__lhs* is equivalent to *__rhs*, false otherwise.

regex_match() [1/7]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_match (
            _Bi_iter __first,
            _Bi_iter __last,
            const basic_regex< _Ch_type, _Rx_traits > & __re,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Indicates if there is a match between the regular expression `e` and all of the character sequence [first, last).

Parameters

__first	Beginning of the character sequence to match.
__last	One-past-the-end of the character sequence to match.
__re	The regular expression.
__flags	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_match().

regex_match() [2/7]

```
template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
bool std::regex_match (
             _Bi_iter __s,
             _Bi_iter __e,
             match_results< _Bi_iter, _Alloc > & __m,
             const basic_regex< _Ch_type, _Rx_traits > & __re,
             regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Determines if there is a match between the regular expression e and all of the character sequence [first, last).

Parameters

__s	Start of the character sequence to match.
__e	One-past-the-end of the character sequence to match.
__m	The match results.
__re	The regular expression.
__flags	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default.
Referenced by regex_match(), regex_match(), regex_match(), regex_match(), and regex_match().

regex_match() [3/7]

```
template<typename _Ch_type, class _Rx_traits>
bool std::regex_match (
            const _Ch_type * __s,
            const basic_regex< _Ch_type, _Rx_traits > & __re,
            regex_constants::match_flag_type __f = regex_constants::match_default)  [inline]
```

Indicates if there is a match between the regular expression e and a C-style null-terminated string.

Parameters

_↩ _s	The C-style null-terminated string to match.
_↩ _re	The regular expression.
_↩ _f	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_match().

regex_match() [4/7]

```
template<typename _Ch_type, typename _Alloc, typename _Rx_traits>
bool std::regex_match (
            const _Ch_type * __s,
            match_results< const _Ch_type *, _Alloc > & __m,
            const basic_regex< _Ch_type, _Rx_traits > & __re,
            regex_constants::match_flag_type __f = regex_constants::match_default)  [inline]
```

Determines if there is a match between the regular expression e and a C-style null-terminated string.

Parameters

_↩ _s	The C-style null-terminated string to match.
_↩ _m	The match results.

_↩ _re	The regular expression.
_↩ _f	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_match().

regex_match() [5/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _↩
Rx_traits>
bool std::regex_match (
             const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > && ,
             match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↩
iterator, _Alloc > & ,
             const basic_regex< _Ch_type, _Rx_traits > & ,
             regex_constants::match_flag_type  = regex_constants::match_default)  [delete]
```

Prevent unsafe attempts to get match_results from a temporary string.
References std::regex_constants::match_default.

regex_match() [6/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _↩
Rx_traits>
bool std::regex_match (
             const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
             match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↩
iterator, _Alloc > & __m,
             const basic_regex< _Ch_type, _Rx_traits > & __re,
             regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Determines if there is a match between the regular expression e and a string.

Parameters

__s	The string to match.

__m	The match results.
__re	The regular expression.
__flags	Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

References std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), std::regex_constants::match_default, and regex_match().

regex_match() [7/7]

```
template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>
bool std::regex_match (
             const basic_string< _Ch_type, _Ch_traits, _Str_allocator > & __s,
             const basic_regex< _Ch_type, _Rx_traits > & __re,
             regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Indicates if there is a match between the regular expression e and a string.

Parameters

__s	[IN] The string to match.
__re	[IN] The regular expression.
__flags	[IN] Controls how the regular expression is matched.

Return values

true	A match exists.
false	Otherwise.

Exceptions

an	exception of type regex_error.

References std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), std::regex_constants::match_default, and regex_match().

regex_replace() [1/6]

```
template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
_Out_iter std::regex_replace (
            _Out_iter __out,
            _Bi_iter __first,
            _Bi_iter __last,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            const _Ch_type * __fmt,
            regex_constants::match_flag_type __flags = regex_constants::match_default)
```

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format C-string.

Parameters

__out	[OUT] The output iterator.
__first	[IN] The start of the string to search.
__last	[IN] One-past-the-end of the string to search.
__e	[IN] The regular expression to search for.
__fmt	[IN] The format C-string.
__flags	[IN] Search and replace policy flags.

Returns

__out

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default.

regex_replace() [2/6]

```
template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename
_St, typename _Sa>
_Out_iter std::regex_replace (
            _Out_iter __out,
            _Bi_iter __first,
            _Bi_iter __last,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            const basic_string< _Ch_type, _St, _Sa > & __fmt,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Search for a regular expression within a range for multiple times, and replace the matched parts through filling a format string.

Parameters

__out	[OUT] The output iterator.
__first	[IN] The start of the string to search.
__last	[IN] One-past-the-end of the string to search.
__e	[IN] The regular expression to search for.
__fmt	[IN] The format string.
__flags	[IN] Search and replace policy flags.

Returns

__out

Exceptions

an	exception of type regex_error.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::length(), and std::regex_constants::match_default.
Referenced by regex_replace(), regex_replace(), regex_replace(), and regex_replace().

regex_replace() [3/6]

```
template<typename _Rx_traits, typename _Ch_type>
basic_string< _Ch_type > std::regex_replace (
             const _Ch_type * __s,
             const basic_regex< _Ch_type, _Rx_traits > & __e,
             const _Ch_type * __fmt,
             regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format C-string.

Parameters

__s	[IN] The C-string to search and replace.
__e	[IN] The regular expression to search for.
__fmt	[IN] The format C-string.
__flags	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

an	exception of type regex_error.

References back_inserter(), std::regex_constants::match_default, and regex_replace().

regex_replace() [4/6]

```
template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
basic_string< _Ch_type > std::regex_replace (
            const _Ch_type * __s,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            const basic_string< _Ch_type, _St, _Sa > & __fmt,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Search for a regular expression within a C-string for multiple times, and replace the matched parts through filling a format string.

Parameters

__s	[IN] The C-string to search and replace.
__e	[IN] The regular expression to search for.
__fmt	[IN] The format string.
__flags	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

an	exception of type regex_error.

References back_inserter(), std::regex_constants::match_default, and regex_replace().

regex_replace() [5/6]

```
template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
basic_string< _Ch_type, _St, _Sa > std::regex_replace (
            const basic_string< _Ch_type, _St, _Sa > & __s,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            const _Ch_type * __fmt,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format C-string.

Parameters

__s	[IN] The string to search and replace.
__e	[IN] The regular expression to search for.
__fmt	[IN] The format C-string.
__flags	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

an	exception of type regex_error.

References back_inserter(), std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), std::regex_constants::match_default, and regex_replace().

regex_replace() [6/6]

```
template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename
_Fsa>
basic_string< _Ch_type, _St, _Sa > std::regex_replace (
            const basic_string< _Ch_type, _St, _Sa > & __s,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            const basic_string< _Ch_type, _Fst, _Fsa > & __fmt,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Search for a regular expression within a string for multiple times, and replace the matched parts through filling a format string.

Parameters

__s	[IN] The string to search and replace.
__e	[IN] The regular expression to search for.
__fmt	[IN] The format string.
__flags	[IN] Search and replace policy flags.

Returns

The string after replacing.

Exceptions

an	exception of type regex_error.

References back_inserter(), std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), std::regex_constants::match_default, and regex_replace().

regex_search() [1/7]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
            _Bi_iter __first,
            _Bi_iter __last,
            const basic_regex< _Ch_type, _Rx_traits > & __re,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Searches for a regular expression within a range.

Parameters

__first	[IN] The start of the string to search.
__last	[IN] One-past-the-end of the string to search.
__re	[IN] The regular expression to search for.
__flags	[IN] Search policy flags.

Return values

true	A match was found within the string.
false	No match was found within the string.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_search().

regex_search() [2/7]

```
template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
            _Bi_iter __s,
            _Bi_iter __e,
            match_results< _Bi_iter, _Alloc > & __m,
            const basic_regex< _Ch_type, _Rx_traits > & __re,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Searches for a regular expression within a range.

Parameters

__s	[IN] The start of the string to search.

__e	[IN] One-past-the-end of the string to search.
__m	[OUT] The match results.
__re	[IN] The regular expression to search for.
__flags	[IN] Search policy flags.

Return values

true	A match was found within the string.
false	No match was found within the string, the content of m is undefined.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default.
Referenced by std::regex_iterator< const char * >::regex_iterator(), std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++(), regex_search(), regex_search(), regex_search(), regex_search(), and regex_search().

regex_search() [3/7]

```
template<typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
            const _Ch_type * __s,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            regex_constants::match_flag_type __f = regex_constants::match_default)  [inline]
```

Searches for a regular expression within a C-string.

Parameters

_↩ _s	[IN] The C-string to search.
_↩ _e	[IN] The regular expression to search for.
_↩ _f	[IN] Search policy flags.

Return values

true	A match was found within the string.

false	No match was found within the string.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_search().

regex_search() [4/7]

```
template<typename _Ch_type, class _Alloc, class _Rx_traits>
bool std::regex_search (
             const _Ch_type * __s,
             match_results< const _Ch_type *, _Alloc > & __m,
             const basic_regex< _Ch_type, _Rx_traits > & __e,
             regex_constants::match_flag_type __f = regex_constants::match_default)  [inline]
```

Searches for a regular expression within a C-string.

Parameters

_↩ _*s*	[IN] A C-string to search for the regex.
_↩ _*m*	[OUT] The set of regex matches.
_↩ _*e*	[IN] The regex to search for in `s`.
_↩ _*f*	[IN] The search flags.

Return values

true	A match was found within the string.
false	No match was found within the string, the content of m is undefined.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_search().

regex_search() [5/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _↩
Rx_traits>
```

```
bool std::regex_search (
            const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > && ,
            match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↩
iterator, _Alloc > & ,
            const basic_regex< _Ch_type, _Rx_traits > & ,
            regex_constants::match_flag_type  = regex_constants::match_default)  [delete]
```

Prevent unsafe attempts to get match_results from a temporary string.
References std::regex_constants::match_default.

regex_search() [6/7]

```
template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _↩
Rx_traits>
bool std::regex_search (
            const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
            match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_↩
iterator, _Alloc > & __m,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            regex_constants::match_flag_type __f = regex_constants::match_default)  [inline]
```

Searches for a regular expression within a string.

Parameters

_↩ _s	[IN] A C++ string to search for the regex.
_↩ _m	[OUT] The set of regex matches.
_↩ _e	[IN] The regex to search for in s.
_↩ _f	[IN] The search flags.

Return values

true	A match was found within the string.
false	No match was found within the string, the content of m is undefined.

Exceptions

an	exception of type regex_error.

References std::basic_string< _CharT, _Traits, _Alloc >::begin(), std::basic_string< _CharT, _Traits, _Alloc >::end(), std::regex_constants::match_default, and regex_search().

regex_search() [7/7]

```
template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>
bool std::regex_search (
            const basic_string< _Ch_type, _Ch_traits, _String_allocator > & __s,
            const basic_regex< _Ch_type, _Rx_traits > & __e,
            regex_constants::match_flag_type __flags = regex_constants::match_default)  [inline]
```

Searches for a regular expression within a string.

Parameters

__s	[IN] The string to search.
__e	[IN] The regular expression to search for.
__flags	[IN] Search policy flags.

Return values

true	A match was found within the string.
false	No match was found within the string.

Exceptions

an	exception of type regex_error.

References std::regex_constants::match_default, and regex_search().

swap() [1/2]

```
template<typename _Ch_type, typename _Rx_traits>
void swap (
            basic_regex< _Ch_type, _Rx_traits > & __lhs,
            basic_regex< _Ch_type, _Rx_traits > & __rhs)  [related]
```

Swaps the contents of two regular expression objects.

Parameters

__lhs	First regular expression.
__rhs	Second regular expression.

swap() [2/2]

```
template<typename _Bi_iter, typename _Alloc>
void swap (
            match_results< _Bi_iter, _Alloc > & __lhs,
            match_results< _Bi_iter, _Alloc > & __rhs)  [related]
```

Swaps two match results.

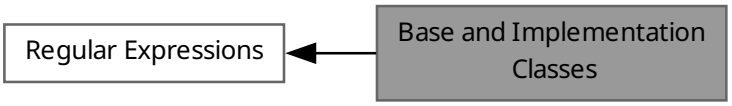
Parameters

__lhs	A match result.
__rhs	A match result.

The contents of the two match_results objects are swapped.

3.13.4 Base and Implementation Classes

Collaboration diagram for Base and Implementation Classes:



Classes

- struct std::__detail::_BracketMatcher< _TraitsT, __icase, __collate >
- class std::__detail::_Compiler< _TraitsT >
- class std::__detail::_Executor< _BiIter, _Alloc, _TraitsT >
- class std::__detail::_Scanner< _CharT >
- class std::__detail::_StateSeq< _TraitsT >

Typedefs

- template<typename _CharT>
  using **std::__detail::_Matcher**
- typedef long **std::__detail::_StateIdT**

Enumerations

- enum std::__detail::_Opcode : int {
  _S_opcode_unknown , **_S_opcode_alternative** , **_S_opcode_repeat** , **_S_opcode_backref** ,
  _S_opcode_line_begin_assertion , **_S_opcode_line_end_assertion** , **_S_opcode_word_boundary** , **_S_↩opcode_subexpr_lookahead** ,
  _S_opcode_subexpr_begin , **_S_opcode_subexpr_end** , **_S_opcode_dummy** , **_S_opcode_match** ,
  _S_opcode_accept }

Variables

- constexpr _StateIdT **std::__detail::_S_invalid_state_id**

3.13.4.1 Detailed Description

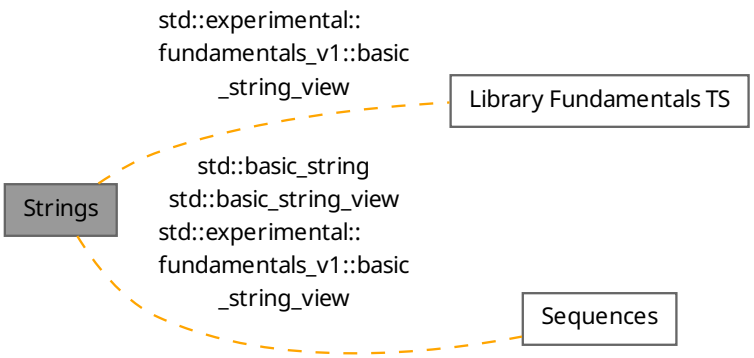
3.13.4.2 Enumeration Type Documentation

_Opcode

`enum std::__detail::_Opcode : int`
Operation codes that define the type of transitions within the base NFA that represents the regular expression.

3.14 Strings

Collaboration diagram for Strings:



Classes

- class std::basic_string< _CharT, _Traits, _Alloc >
- class std::basic_string_view< _CharT, _Traits >
- class std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >
- struct std::char_traits< _CharT >

Typedefs

- typedef basic_string< char > std::string
- typedef basic_string< char16_t > std::u16string
- typedef basic_string< char32_t > std::u32string
- typedef basic_string< wchar_t > std::wstring

3.14.1 Detailed Description

3.14.2 Typedef Documentation

string

`typedef basic_string<char> std::string`
A string of `char`.

u16string

```
typedef basic_string<char16_t> std::u16string
```
A string of `char16_t`.

u32string

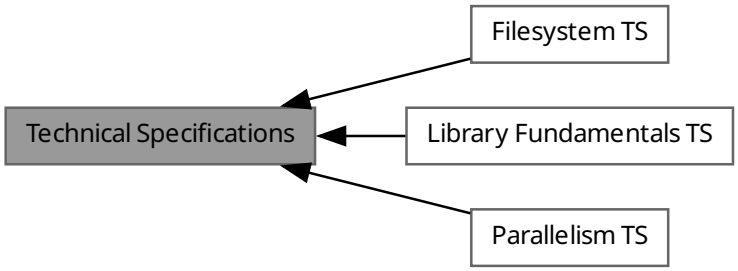
```
typedef basic_string<char32_t> std::u32string
```
A string of `char32_t`.

wstring

```
typedef basic_string<wchar_t> std::wstring
```
A string of `wchar_t`.

3.15 Technical Specifications

Collaboration diagram for Technical Specifications:



Topics

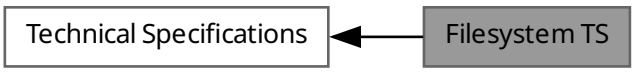
- Filesystem TS
- Library Fundamentals TS
- Parallelism TS

3.15.1 Detailed Description

Components specified by various Technical Specifications.
As indicated by the std::experimental namespace and the header paths, the contents of these Technical Specifications are experimental and not part of the C++ standard. As such the interfaces and implementations may change in the future, and there is **no guarantee of compatibility between different GCC releases** for these features.

3.15.2 Filesystem TS

Collaboration diagram for Filesystem TS:



Files

- file filesystem

Classes

- class std::experimental::filesystem::v1::filesystem_error
- class std::experimental::filesystem::v1::path::iterator
- class std::experimental::filesystem::v1::path
- struct std::experimental::filesystem::v1::space_info

Typedefs

- using std::experimental::filesystem::file_time_type

Enumerations

- enum class std::experimental::filesystem::copy_options : unsigned short {
  none , **skip_existing** , **overwrite_existing** , **update_existing** ,
  recursive , **copy_symlinks** , **skip_symlinks** , **directories_only** ,
  create_symlinks , **create_hard_links** }
- enum class std::experimental::filesystem::directory_options : unsigned char { **none** , **follow_directory_symlink** , **skip_permission_denied** }
- enum class std::experimental::filesystem::file_type : signed char {
  none , **not_found** , **regular** , **directory** ,
  symlink , **block** , **character** , **fifo** ,
  socket , **unknown** }
- enum class std::experimental::filesystem::perms : unsigned {
  none , **owner_read** , **owner_write** , **owner_exec** ,
  owner_all , **group_read** , **group_write** , **group_exec** ,
  group_all , **others_read** , **others_write** , **others_exec** ,
  others_all , **all** , **set_uid** , **set_gid** ,
  sticky_bit , **mask** , **unknown** , **add_perms** ,
  remove_perms , **symlink_nofollow** }

Functions

- **std::experimental::filesystem::v1::path::path** (const path &__p)
- **std::experimental::filesystem::v1::path::path** (path &&__p) noexcept
- **std::experimental::filesystem::v1::path::path** (string_type &&__source)

- path **std::experimental::filesystem::absolute** (const path &__p, const path &__base=current_path())
- path & **std::experimental::filesystem::v1::path::assign** (string_type &&__source)
- directory_iterator **std::experimental::filesystem::begin** (directory_iterator __iter) noexcept
- recursive_directory_iterator **std::experimental::filesystem::begin** (recursive_directory_iterator __iter) noexcept
- iterator **std::experimental::filesystem::v1::path::begin** () const noexcept
- path **std::experimental::filesystem::canonical** (const path &__p, const path &__base, error_code &__ec)
- path **std::experimental::filesystem::canonical** (const path &__p, const path &__base=current_path())
- path **std::experimental::filesystem::canonical** (const path &__p, error_code &__ec)
- int **std::experimental::filesystem::v1::path::compare** (const basic_string_view< value_type > __s) const
- int **std::experimental::filesystem::v1::path::compare** (const string_type &__s) const
- int **std::experimental::filesystem::v1::path::compare** (const value_type *__s) const
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to)
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, copy_options __options)
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, copy_options __options, error_code &) noexcept
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, error_code &__ec) noexcept
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to)
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option, error_code &)
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, error_code &__ec)
- void **std::experimental::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_↩ symlink)
- void **std::experimental::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_↩ symlink, error_code &__ec) noexcept
- bool **std::experimental::filesystem::create_directories** (const path &__p)
- bool **std::experimental::filesystem::create_directories** (const path &__p, error_code &__ec)
- bool **std::experimental::filesystem::create_directory** (const path &__p)
- bool **std::experimental::filesystem::create_directory** (const path &__p, const path &__attributes)
- bool **std::experimental::filesystem::create_directory** (const path &__p, const path &__attributes, error_code &__ec) noexcept
- bool **std::experimental::filesystem::create_directory** (const path &__p, error_code &__ec) noexcept
- void **std::experimental::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink)
- void **std::experimental::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- void **std::experimental::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link)
- void **std::experimental::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link, error_code &__ec) noexcept
- void **std::experimental::filesystem::create_symlink** (const path &__to, const path &__new_symlink)
- void **std::experimental::filesystem::create_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- path **std::experimental::filesystem::current_path** ()
- void **std::experimental::filesystem::current_path** (const path &__p)
- void **std::experimental::filesystem::current_path** (const path &__p, error_code &__ec) noexcept
- path **std::experimental::filesystem::current_path** (error_code &__ec)
- directory_iterator **std::experimental::filesystem::end** (directory_iterator) noexcept
- recursive_directory_iterator **std::experimental::filesystem::end** (recursive_directory_iterator) noexcept
- iterator **std::experimental::filesystem::v1::path::end** () const noexcept
- bool **std::experimental::filesystem::equivalent** (const path &__p1, const path &__p2)
- bool **std::experimental::filesystem::equivalent** (const path &__p1, const path &__p2, error_code &__ec) noexcept

- bool **std::experimental::filesystem::exists** (const path &__p)
- bool **std::experimental::filesystem::exists** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::exists** (file_status __s) noexcept
- path **std::experimental::filesystem::v1::path::extension** () const
- uintmax_t **std::experimental::filesystem::file_size** (const path &__p)
- uintmax_t **std::experimental::filesystem::file_size** (const path &__p, error_code &__ec) noexcept
- path **std::experimental::filesystem::v1::path::filename** () const
- std::string **std::experimental::filesystem::v1::path::generic_string** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> std::basic_string< _CharT, _Traits, _Allocator > **std::experimental::filesystem::v1::path::generic_string** (const _Allocator &__a=_Allocator()) const
- std::u16string **std::experimental::filesystem::v1::path::generic_u16string** () const
- std::u32string **std::experimental::filesystem::v1::path::generic_u32string** () const
- std::string **std::experimental::filesystem::v1::path::generic_u8string** () const
- std::wstring **std::experimental::filesystem::v1::path::generic_wstring** () const
- uintmax_t **std::experimental::filesystem::hard_link_count** (const path &__p)
- uintmax_t **std::experimental::filesystem::hard_link_count** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::v1::path::has_extension** () const
- bool **std::experimental::filesystem::v1::path::has_stem** () const
- bool **std::experimental::filesystem::v1::path::is_absolute** () const
- bool **std::experimental::filesystem::is_block_file** (const path &__p)
- bool **std::experimental::filesystem::is_block_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_block_file** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_character_file** (const path &__p)
- bool **std::experimental::filesystem::is_character_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_character_file** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_directory** (const path &__p)
- bool **std::experimental::filesystem::is_directory** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_directory** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_empty** (const path &__p)
- bool **std::experimental::filesystem::is_empty** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_fifo** (const path &__p)
- bool **std::experimental::filesystem::is_fifo** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_fifo** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_other** (const path &__p)
- bool **std::experimental::filesystem::is_other** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_other** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_regular_file** (const path &__p)
- bool **std::experimental::filesystem::is_regular_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_regular_file** (file_status) noexcept
- bool **std::experimental::filesystem::is_socket** (const path &__p)
- bool **std::experimental::filesystem::is_socket** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_socket** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_symlink** (const path &__p)
- bool **std::experimental::filesystem::is_symlink** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_symlink** (file_status) noexcept
- file_time_type **std::experimental::filesystem::last_write_time** (const path &__p)
- file_time_type **std::experimental::filesystem::last_write_time** (const path &__p, error_code &__ec) noexcept
- void **std::experimental::filesystem::last_write_time** (const path &__p, file_time_type __new_time)
- void **std::experimental::filesystem::last_write_time** (const path &__p, file_time_type __new_time, error_code &__ec) noexcept

- path & **std::experimental::filesystem::v1::path::make_preferred** ()
- bool **std::experimental::filesystem::operator!=** (const directory_iterator &__lhs, const directory_iterator &__rhs)
- bool **std::experimental::filesystem::operator!=** (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs)
- copy_options & **std::experimental::filesystem::operator&=** (copy_options &__x, copy_options __y) noexcept
- reference **std::experimental::filesystem::v1::path::iterator::operator*** () const noexcept
- iterator & **std::experimental::filesystem::v1::path::iterator::operator++** () noexcept
- template<typename _CharT> __detail::_Path< _CharT *, _CharT * > & **std::experimental::filesystem::v1::path::operator+=** (_CharT __x)
- path & **std::experimental::filesystem::v1::path::operator+=** (basic_string_view< value_type > __x)
- path & **std::experimental::filesystem::v1::path::operator+=** (const path &__x)
- path & **std::experimental::filesystem::v1::path::operator+=** (const string_type &__x)
- path & **std::experimental::filesystem::v1::path::operator+=** (const value_type *__x)
- path & **std::experimental::filesystem::v1::path::operator+=** (value_type __x)
- iterator & **std::experimental::filesystem::v1::path::iterator::operator--** () noexcept
- bool **std::experimental::filesystem::operator<** (const path &__lhs, const path &__rhs) noexcept
- path & **std::experimental::filesystem::v1::path::operator=** (const path &__p)
- path & **std::experimental::filesystem::v1::path::operator=** (path &&__p) noexcept
- path & **std::experimental::filesystem::v1::path::operator=** (string_type &&__source)
- bool **std::experimental::filesystem::operator==** (const directory_iterator &__lhs, const directory_iterator &__rhs)
- bool **std::experimental::filesystem::operator==** (const path &__lhs, const path &__rhs) noexcept
- bool **std::experimental::filesystem::operator==** (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs)
- constexpr copy_options **std::experimental::filesystem::operator^** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::experimental::filesystem::operator^=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::experimental::filesystem::operator|** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::experimental::filesystem::operator|=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::experimental::filesystem::operator~** (copy_options __x) noexcept
- void **std::experimental::filesystem::permissions** (const path &__p, perms __prms)
- void **std::experimental::filesystem::permissions** (const path &__p, perms __prms, error_code &__ec) noexcept
- path **std::experimental::filesystem::read_symlink** (const path &__p)
- path **std::experimental::filesystem::read_symlink** (const path &__p, error_code &__ec)
- bool **std::experimental::filesystem::remove** (const path &__p)
- bool **std::experimental::filesystem::remove** (const path &__p, error_code &__ec) noexcept
- uintmax_t **std::experimental::filesystem::remove_all** (const path &__p)
- uintmax_t **std::experimental::filesystem::remove_all** (const path &__p, error_code &__ec)
- void **std::experimental::filesystem::rename** (const path &__from, const path &__to)
- void **std::experimental::filesystem::rename** (const path &__from, const path &__to, error_code &__ec) noexcept
- void **std::experimental::filesystem::resize_file** (const path &__p, uintmax_t __size)
- void **std::experimental::filesystem::resize_file** (const path &__p, uintmax_t __size, error_code &__ec) noexcept
- space_info **std::experimental::filesystem::space** (const path &__p)
- space_info **std::experimental::filesystem::space** (const path &__p, error_code &__ec) noexcept
- file_status **std::experimental::filesystem::status** (const path &)
- file_status **std::experimental::filesystem::status** (const path &, error_code &) noexcept

- bool **std::experimental::filesystem::status_known** (file_status) noexcept
- path **std::experimental::filesystem::v1::path::stem** () const
- std::string **std::experimental::filesystem::v1::path::string** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>> std::basic_string< _CharT, _Traits, _Allocator > **std::experimental::filesystem::v1::path::string** (const _↩ Allocator &__a=_Allocator()) const
- void **std::experimental::filesystem::v1::path::swap** (path &__rhs) noexcept
- file_status **std::experimental::filesystem::symlink_status** (const path &)
- file_status **std::experimental::filesystem::symlink_status** (const path &, error_code &) noexcept
- path **std::experimental::filesystem::system_complete** (const path &__p)
- path **std::experimental::filesystem::system_complete** (const path &__p, error_code &__ec)
- path **std::experimental::filesystem::temp_directory_path** ()
- path **std::experimental::filesystem::temp_directory_path** (error_code &__ec)
- std::u16string **std::experimental::filesystem::v1::path::u16string** () const
- std::u32string **std::experimental::filesystem::v1::path::u32string** () const
- std::string **std::experimental::filesystem::v1::path::u8string** () const
- std::wstring **std::experimental::filesystem::v1::path::wstring** () const

- constexpr perms **std::experimental::filesystem::operator|** (perms __x, perms __y) noexcept
- constexpr perms **std::experimental::filesystem::operator^** (perms __x, perms __y) noexcept
- constexpr perms **std::experimental::filesystem::operator~** (perms __x) noexcept
- perms & **std::experimental::filesystem::operator&=** (perms &__x, perms __y) noexcept
- perms & **std::experimental::filesystem::operator|=** (perms &__x, perms __y) noexcept
- perms & **std::experimental::filesystem::operator^=** (perms &__x, perms __y) noexcept

- constexpr directory_options **std::experimental::filesystem::operator|** (directory_options __x, directory_options __y) noexcept
- constexpr directory_options **std::experimental::filesystem::operator^** (directory_options __x, directory_↩ options __y) noexcept
- constexpr directory_options **std::experimental::filesystem::operator~** (directory_options __x) noexcept
- directory_options & **std::experimental::filesystem::operator&=** (directory_options &__x, directory_options _↩ _y) noexcept
- directory_options & **std::experimental::filesystem::operator|=** (directory_options &__x, directory_options __y) noexcept
- directory_options & **std::experimental::filesystem::operator^=** (directory_options &__x, directory_options _↩ _y) noexcept

3.15.2.1 Detailed Description

Utilities for performing operations on file systems and their components, such as paths, regular files, and directories. ISO/IEC TS 18822:2015 C++ File System Technical Specification

Since

C++11

Remarks

Link using `-lstdc++fs` to use these types and functions.

3.15.2.2 Typedef Documentation

file_time_type

```
using std::experimental::filesystem::v1::file_time_type
```

The type used for file timestamps.

3.15.2.3 Enumeration Type Documentation

copy_options

`enum class std::experimental::filesystem::v1::copy_options : unsigned short [strong]`
Bitmask type controlling effects of `filesystem::copy`.

directory_options

`enum class std::experimental::filesystem::v1::directory_options : unsigned char [strong]`
Bitmask type controlling directory iteration.

file_type

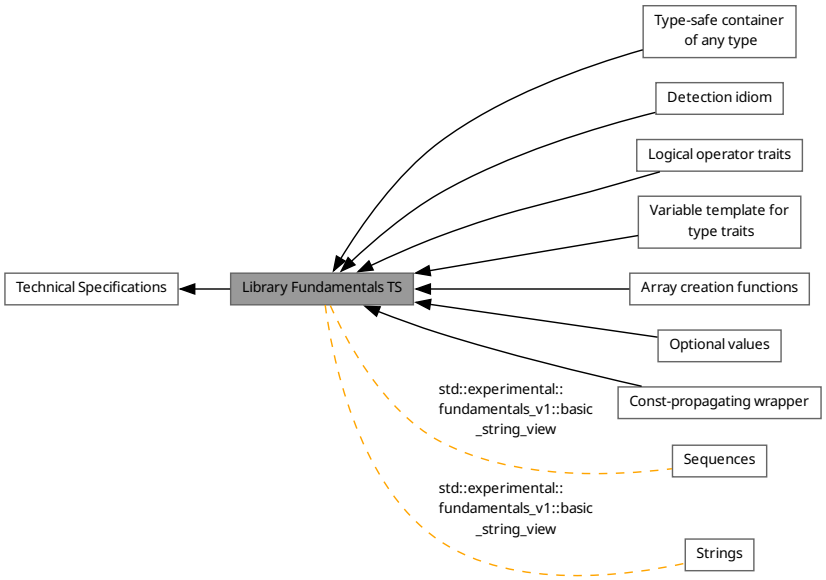
`enum class std::experimental::filesystem::v1::file_type : signed char [strong]`
Enumerated type representing the type of a file.

perms

`enum class std::experimental::filesystem::v1::perms : unsigned [strong]`
Bitmask type representing file access permissions.

3.15.3 Library Fundamentals TS

Collaboration diagram for Library Fundamentals TS:



Topics

- Array creation functions
- Const-propagating wrapper
- Detection idiom
- Logical operator traits

- Optional values
- Type-safe container of any type
- Variable template for type traits

Files

- file algorithm
- file any
- file array
- file chrono
- file deque
- file forward_list
- file functional
- file iterator
- file list
- file map
- file memory
- file memory_resource
- file numeric
- file optional
- file random
- file ratio
- file regex
- file set
- file string
- file string_view
- file system_error
- file tuple
- file type_traits
- file unordered_map
- file unordered_set
- file utility
- file vector
- file propagate_const

Classes

- class std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >

3.15.3.1 Detailed Description

Components defined by the *C++ Extensions for Library Fundamentals* Technical Specification, versions 1 and 2.

- ISO/IEC TS 19568:2015 C++ Extensions for Library Fundamentals
- ISO/IEC TS 19568:2017 C++ Extensions for Library Fundamentals, Version 2

3.15.3.2 Array creation functions

Collaboration diagram for Array creation functions:



Functions

- template<typename _Tp, size_t _Nm, size_t... _Idx>
  constexpr array< remove_cv_t< _Tp >, _Nm > **std::experimental::__to_array** (_Tp(&__a)[_Nm]↩
  , index_sequence< _Idx... >)
- template<typename _Dest = void, typename... _Types>
  constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> std::experimental::make_array
  (_Types &&... __t)
- template<typename _Tp, size_t _Nm>
  constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::to_array (_Tp(&__a)[_Nm]) noexcept(is_nothrow_constructible<
  remove_cv_t< _Tp >, _Tp & >::value)

3.15.3.2.1 Detailed Description

Array creation functions as described in N4529, Working Draft, C++ Extensions for Library Fundamentals, Version 2

3.15.3.2.2 Function Documentation

make_array()

```
template<typename _Dest = void, typename...  _Types>
array< typename __make_array_elem< _Dest, _Types...  >::type, sizeof...(_Types)> std::experimental::↩
fundamentals_v2::make_array (
            _Types &&...  __t)  [constexpr]
```

Create a std::array from a variable-length list of arguments.
References std::forward().

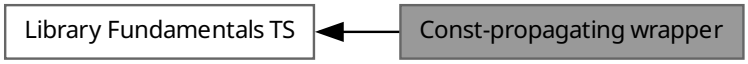
to_array()

```
template<typename _Tp, size_t _Nm>
array< remove_cv_t< _Tp >, _Nm > std::experimental::fundamentals_v2::to_array (
            _Tp(&) __a[_Nm])  [constexpr], [noexcept]
```

Create a std::array from an array.

3.15.3.3 Const-propagating wrapper

Collaboration diagram for Const-propagating wrapper:



Classes

- class std::experimental::fundamentals_v2::propagate_const< _Tp >

Functions

- template<typename _Tp>
  constexpr const _Tp & **std::experimental::get_underlying** (const propagate_const< _Tp > &__pt) noexcept
- template<typename _Tp>
  constexpr _Tp & **std::experimental::get_underlying** (propagate_const< _Tp > &__pt) noexcept
- constexpr **std::experimental::fundamentals_v2::__propagate_const_conversion_nc< _Tp, _Elem, true >::operator _Elem** * ()
- constexpr **std::experimental::fundamentals_v2::__propagate_const_conversions< _Tp * >::operator _Tp** * () noexcept
- constexpr **std::experimental::fundamentals_v2::__propagate_const_conversion_c< _Tp, _Elem, true >::operator const _Elem** * () const
- constexpr **std::experimental::fundamentals_v2::__propagate_const_conversions< _Tp * >::operator const _Tp** * () const noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr bool **std::experimental::operator!=** (const propagate_const< _Tp > &__pt, nullptr_t)
- template<typename _Tp>
  constexpr bool **std::experimental::operator!=** (nullptr_t, const propagate_const< _Tp > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**< (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**< (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**< (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator<=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator<=** (const propagate_const< _Tp > &__pt, const _Up &__u)

- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator<=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr bool **std::experimental::operator==** (const propagate_const< _Tp > &__pt, nullptr_t)
- template<typename _Tp>
  constexpr bool **std::experimental::operator==** (nullptr_t, const propagate_const< _Tp > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr enable_if_t< __is_swappable< _Tp >::value, void > **std::experimental::swap** (propagate_const< _Tp > &__pt, propagate_const< _Tp > &__pt2) noexcept(__is_nothrow_swappable< _Tp >::value)

3.15.3.3.1 Detailed Description

A const-propagating wrapper that propagates const to pointer-like members, as described in n4388 "A Proposal to Add a Const-Propagating Wrapper to the Standard Library".

3.15.3.4 Detection idiom

Collaboration diagram for Detection idiom:



- #define __cpp_lib_experimental_detect

- template<typename...>
  using std::experimental::void_t
- template<typename _Default, template< typename... > class _Op, typename... _Args>
  using std::experimental::detected_or
- template<typename _Default, template< typename... > class _Op, typename... _Args>
  using std::experimental::detected_or_t
- template<template< typename... > class _Op, typename... _Args>
  using std::experimental::detected_t
- template<template< typename... > class _Op, typename... _Args>
  using std::experimental::is_detected
- template<typename _Expected, template< typename... > class _Op, typename... _Args>
  using std::experimental::is_detected_exact
- template<typename _To, template< typename... > class _Op, typename... _Args>
  using std::experimental::is_detected_convertible
- template<template< typename... > class _Op, typename... _Args>
  constexpr bool std::experimental::is_detected_v
- template<typename _Expected, template< typename... > class _Op, typename... _Args>
  constexpr bool std::experimental::is_detected_exact_v
- template<typename _To, template< typename... > class _Op, typename... _Args>
  constexpr bool std::experimental::is_detected_convertible_v

3.15.3.4.1 Detailed Description

Since

Library Fundamentals TS v2. C++14.

3.15.3.4.2 Macro Definition Documentation

__cpp_lib_experimental_detect

```
#define __cpp_lib_experimental_detect
```

A metafunction that always yields void, used for detecting valid types.

3.15.3.4.3 Typedef Documentation

detected_or

```
template<typename _Default, template< typename...  > class _Op, typename...  _Args>
using std::experimental::fundamentals_v2::detected_or
```

A metafunction that always yields void, used for detecting valid types.

detected_or_t

```
template<typename _Default, template< typename...  > class _Op, typename...  _Args>
using std::experimental::fundamentals_v2::detected_or_t
```

A metafunction that always yields void, used for detecting valid types.

detected_t

```
template<template< typename...  > class _Op, typename...  _Args>
using std::experimental::fundamentals_v2::detected_t
```

A metafunction that always yields void, used for detecting valid types.

is_detected

```
template<template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::is_detected
```
A metafunction that always yields void, used for detecting valid types.

is_detected_convertible

```
template<typename _To, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::is_detected_convertible
```
A metafunction that always yields void, used for detecting valid types.

is_detected_exact

```
template<typename _Expected, template< typename... > class _Op, typename... _Args>
using std::experimental::fundamentals_v2::is_detected_exact
```
A metafunction that always yields void, used for detecting valid types.

void_t

```
template<typename...>
using std::experimental::fundamentals_v2::void_t
```
A metafunction that always yields void, used for detecting valid types.

3.15.3.4.4 Variable Documentation

is_detected_convertible_v

```
template<typename _To, template< typename... > class _Op, typename... _Args>
bool std::experimental::fundamentals_v2::is_detected_convertible_v [constexpr]
```
A metafunction that always yields void, used for detecting valid types.

is_detected_exact_v

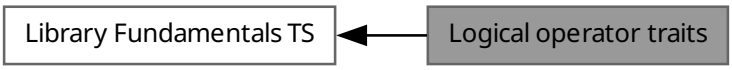
```
template<typename _Expected, template< typename... > class _Op, typename... _Args>
bool std::experimental::fundamentals_v2::is_detected_exact_v [constexpr]
```
A metafunction that always yields void, used for detecting valid types.

is_detected_v

```
template<template< typename... > class _Op, typename... _Args>
bool std::experimental::fundamentals_v2::is_detected_v [constexpr]
```
A metafunction that always yields void, used for detecting valid types.

3.15.3.5 Logical operator traits

Collaboration diagram for Logical operator traits:



- #define **__cpp_lib_experimental_logical_traits**
- template<typename... _Bn>
  constexpr bool **std::experimental::conjunction_v**
- template<typename... _Bn>
  constexpr bool **std::experimental::disjunction_v**
- template<typename _Pp>
  constexpr bool **std::experimental::negation_v**

3.15.3.5.1 Detailed Description

Since

Library Fundamentals TS v2. C++14.

3.15.3.6 Optional values

Collaboration diagram for Optional values:



Classes

- class std::experimental::fundamentals_v1::bad_optional_access
- struct std::experimental::fundamentals_v1::in_place_t
- struct std::experimental::fundamentals_v1::nullopt_t
- class std::experimental::fundamentals_v1::optional< _Tp >

Macros

- #define **__cpp_lib_experimental_optional**

Variables

- constexpr in_place_t std::experimental::in_place
- constexpr nullopt_t std::experimental::nullopt

3.15.3.6.1 Detailed Description

Class template for optional values and surrounding facilities, as described in n3793 "A proposal to add a utility class to represent optional objects (Revision 5)".

3.15.3.6.2 Variable Documentation

in_place

```
in_place_t std::experimental::fundamentals_v1::in_place  [constexpr]
```

Tag for in-place construction.

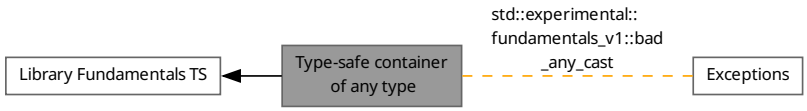
nullopt

```
nullopt_t std::experimental::fundamentals_v1::nullopt  [constexpr]
```

Tag to disengage optional objects.

3.15.3.7 Type-safe container of any type

Collaboration diagram for Type-safe container of any type:



Classes

- class std::experimental::fundamentals_v1::any
- class std::experimental::fundamentals_v1::bad_any_cast

Macros

- #define **__cpp_lib_experimental_any**

Functions

- static void **std::experimental::fundamentals_v1::any::_Manager_external< _Tp >::_S_manage** (_Op __↩ which, const any *__anyp, _Arg *__arg)
- static void **std::experimental::fundamentals_v1::any::_Manager_internal< _Tp >::_S_manage** (_Op __↩ which, const any *__anyp, _Arg *__arg)
- template<typename _ValueType>
  _ValueType std::experimental::any_cast (const any &__any)
- void std::experimental::swap (any &__x, any &__y) noexcept
- template<typename _ValueType>
  _ValueType std::experimental::any_cast (any &__any)
- template<typename _ValueType, typename enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>
  _ValueType std::experimental::any_cast (any &&__any)

- template<typename _ValueType>
  const _ValueType * std::experimental::any_cast (const any *__any) noexcept
- template<typename _ValueType>
  _ValueType * std::experimental::any_cast (any *__any) noexcept

3.15.3.7.1 Detailed Description

A type-safe container for single values of value types, as described in n3804 "Any Library Proposal (Revision 3)".

3.15.3.7.2 Function Documentation

any_cast() [1/5]

```
template<typename _ValueType, typename enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference
_ValueType >::value, bool >::type = true>
_ValueType std::experimental::fundamentals_v1::any_cast (
             any && __any)  [inline]
```

Access the contained object.

Template Parameters

_ValueType	A reference or CopyConstructible type.

Parameters

__any	The object to access.

Returns

The contained object.

Exceptions

bad_any_cast	If `__any.type() != typeid(remove_reference_t<_ValueType>)`

References any_cast().

any_cast() [2/5]

```
template<typename _ValueType>
_ValueType std::experimental::fundamentals_v1::any_cast (
             any & __any)  [inline]
```

Access the contained object.

Template Parameters

_ValueType	A reference or CopyConstructible type.

Parameters

__any	The object to access.

Returns

The contained object.

Exceptions

bad_any_cast	If `__any.type() != typeid(remove_reference_t<_ValueType>)`

References any_cast().

any_cast() [3/5]

```
template<typename _ValueType>
_ValueType * std::experimental::fundamentals_v1::any_cast (
             any * __any)  [inline], [noexcept]
```

Access the contained object.

Template Parameters

_ValueType	The type of the contained object.

Parameters

__any	A pointer to the object to access.

Returns

The address of the contained object if `__any != nullptr && __any.type() == typeid(_ValueType)` , otherwise a null pointer.

any_cast() [4/5]

```
template<typename _ValueType>
_ValueType std::experimental::fundamentals_v1::any_cast (
             const any & __any)  [inline]
```

Access the contained object.

Template Parameters

_ValueType	A const-reference or CopyConstructible type.

Parameters

__any	The object to access.

Returns

The contained object.

Exceptions

bad_any_cast	If `__any.type() != typeid(remove_reference_t<_ValueType>)`

References any_cast().
Referenced by any_cast(), any_cast(), and any_cast().

any_cast() [5/5]

```
template<typename _ValueType>
const _ValueType * std::experimental::fundamentals_v1::any_cast (
            const any * __any)  [inline], [noexcept]
```

Access the contained object.

Template Parameters

_ValueType	The type of the contained object.

Parameters

__any	A pointer to the object to access.

Returns

The address of the contained object if `__any != nullptr && __any.type() == typeid(_↩ ValueType)` , otherwise a null pointer.

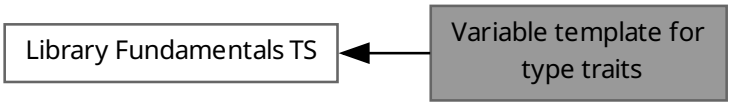
swap()

```
void std::experimental::fundamentals_v1::swap (
            any & __x,
            any & __y)  [inline], [noexcept]
```

Exchange the states of two `any` objects.

3.15.3.8 Variable template for type traits

Collaboration diagram for Variable template for type traits:



- #define **__cpp_lib_experimental_type_trait_variable_templates**
- template<typename _Tp>
  constexpr bool **std::experimental::is_void_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_null_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_integral_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_floating_point_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_array_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_lvalue_reference_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_rvalue_reference_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_member_object_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_member_function_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_enum_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_union_v**

- template<typename _Tp>
  constexpr bool **std::experimental::is_class_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_function_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_reference_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_arithmetic_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_fundamental_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_object_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_scalar_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_compound_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_member_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_const_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_volatile_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivial_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_copyable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_standard_layout_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_pod_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_literal_type_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_empty_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_polymorphic_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_abstract_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_final_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_signed_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_unsigned_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **std::experimental::is_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_default_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_move_constructible_v**

- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_move_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_destructible_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **std::experimental::is_trivially_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_default_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_move_constructible_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_trivially_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_move_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_destructible_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **std::experimental::is_nothrow_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_default_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_move_constructible_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_nothrow_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_move_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_destructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::has_virtual_destructor_v**
- template<typename _Tp>
  constexpr size_t **std::experimental::alignment_of_v**
- template<typename _Tp>
  constexpr size_t **std::experimental::rank_v**
- template<typename _Tp, unsigned _Idx = 0>
  constexpr size_t **std::experimental::extent_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_same_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_same_v< _Tp, _Tp >**

- template<typename _Base, typename _Derived>
  constexpr bool **std::experimental::is_base_of_v**
- template<typename _From, typename _To>
  constexpr bool **std::experimental::is_convertible_v**

3.15.3.8.1 Detailed Description

Since

Library Fundamentals TS v1. C++14.

See also

variable_templates

3.15.4 Parallelism TS

Collaboration diagram for Parallelism TS:



Topics

- Data parallel extensions

3.15.4.1 Detailed Description

Components defined by the *C++ Extensions for Parallelism* Technical Specification.

- ISO/IEC TS 19570:2015 C++ Extensions for Parallelism
- ISO/IEC TS 19570:2018 C++ Extensions for Parallelism, Version 2

3.15.4.2 Data parallel extensions

Collaboration diagram for Data parallel extensions:



Macros

- #define **__cpp_lib_experimental_parallel_simd**

3.15.4.2.1 Detailed Description

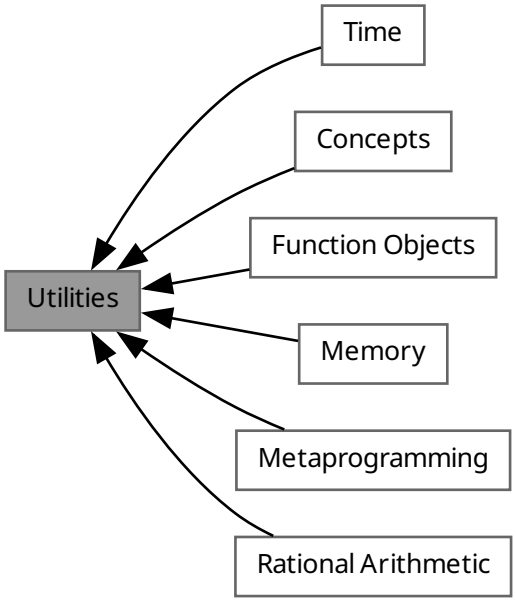
Data-parallel types library.

Since

C++17

3.16 Utilities

Collaboration diagram for Utilities:



Topics

- Concepts
- Function Objects
- Memory
- Metaprogramming
- Rational Arithmetic
- Time

Classes

- class std::bitset< _Nb >
- struct std::pair< _T1, _T2 >
- struct std::piecewise_construct_t
- class std::tuple< _Elements >
- class std::tuple< _T1, _T2 >

- struct std::tuple_element< __i, tuple< _Types... > >
- struct std::tuple_size< tuple< _Elements... > >
- struct std::type_index
- struct std::uses_allocator< tuple< _Types... >, _Alloc >

Typedefs

- template<typename _Tp, typename _Up>
  using **std::__like_t**

Functions

- template<typename... _Args1, typename... _Args2>
  constexpr std::pair< _T1, _T2 >::pair (piecewise_construct_t, tuple< _Args1... >, tuple< _Args2... >)
- template<typename _Tp>
  constexpr _Tp * std::__addressof (_Tp &__r) noexcept
- template<typename _Tp, typename _Up = _Tp>
  constexpr _Tp **std::__exchange** (_Tp &__obj, _Up &&__new_val)
- template<size_t __i, typename _Head, typename... _Tail>
  constexpr _Head & **std::__get_helper** (_Tuple_impl< __i, _Head, _Tail... > &__t) noexcept
- template<size_t __i, typename _Head, typename... _Tail>
  constexpr const _Head & **std::__get_helper** (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept
- template<size_t __i, typename... _Types>
  __enable_if_t<(__i >=sizeof...(_Types))> **std::__get_helper** (const tuple< _Types... > &)=delete
- template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>
  constexpr _Up && **std::__invfwd** (typename remove_reference< _Tp >::type &__t) noexcept
- template<typename _Callable, typename... _Args>
  constexpr __invoke_result< _Callable, _Args... >::type std::__invoke (_Callable &&__fn, _Args &&... __args) noexcept(__is_nothrow_invocable< _Callable, _Args... >::value)
- template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>
  constexpr _Res **std::__invoke_impl** (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __↩
  args)
- template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>
  constexpr _Res **std::__invoke_impl** (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)
- template<typename _Res, typename _MemPtr, typename _Tp>
  constexpr _Res **std::__invoke_impl** (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)
- template<typename _Res, typename _MemPtr, typename _Tp>
  constexpr _Res **std::__invoke_impl** (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)
- template<typename _Res, typename _Fn, typename... _Args>
  constexpr _Res **std::__invoke_impl** (__invoke_other, _Fn &&__f, _Args &&... __args)
- template<typename _Res, typename _Callable, typename... _Args>
  constexpr enable_if_t< is_invocable_r_v< _Res, _Callable, _Args... >, _Res > **std::__invoke_r** (_Callable &&__fn, _Args &&... __args) noexcept(is_nothrow_invocable_r_v< _Res, _Callable, _Args... >)
- template<typename _Cat, typename _Tp, typename _Up, typename _IndexSeq>
  constexpr _Cat **std::__tuple_cmp** (const _Tp &__t, const _Up &__u, _IndexSeq __indices)
- template<typename _Tp>
  constexpr _Tp * std::addressof (_Tp &__r) noexcept
- template<typename _Tp>
  const _Tp * **std::addressof** (const _Tp &&)=delete
- template<typename _Tp>
  auto std::declval () noexcept -> decltype(__declval< _Tp >(0))
- template<typename _Tp>
  constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &&__t) noexcept

- template<typename _Tp>
  constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &__t) noexcept
- template<typename... _Elements>
  constexpr tuple< _Elements &&... > std::forward_as_tuple (_Elements &&... __args) noexcept
- template<typename _Tp, typename _Up>
  constexpr __like_t< _Tp, _Up > std::forward_like (_Up &&__x) noexcept
- template<size_t __i, typename... _Elements>
  constexpr const __tuple_element_t< __i, tuple< _Elements... > > && std::get (const tuple< _Elements... > &&__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr const __tuple_element_t< __i, tuple< _Elements... > > & std::get (const tuple< _Elements... > &__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr __tuple_element_t< __i, tuple< _Elements... > > && std::get (tuple< _Elements... > &&__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr __tuple_element_t< __i, tuple< _Elements... > > & std::get (tuple< _Elements... > &__t) noexcept
- template<typename _T1, typename _T2>
  constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > std::make_pair (_T1 &&__x, _T2 &&__y)
- template<typename... _Elements>
  constexpr tuple< typename __decay_and_strip< _Elements >::__type... > std::make_tuple (_Elements &&... __args)
- template<typename _Tp>
  constexpr std::remove_reference< _Tp >::type && std::move (_Tp &&__t) noexcept
- template<typename _Tp>
  constexpr __conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > std::move_if_noexcept (_Tp &__x) noexcept
- template<typename... _Tps, typename... _Ups>
  constexpr common_comparison_category_t< __detail::__synth3way_t< _Tps, _Ups >... > **std::operator<=>** (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)
- template<typename... _Tps, typename... _Ups>
  constexpr bool **std::operator==** (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)
- template<typename _Tp>
  constexpr void std::swap (_Tp &__a, _Tp &__b) noexcept(/*conditional */) is_nothrow_move_assignable< _Tp > >
- template<typename _Tp, size_t _Nm>
  constexpr void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(/*conditional */)
- template<typename... _Elements>
  constexpr enable_if< __and_< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename... _Elements>
  constexpr tuple< _Elements &... > std::tie (_Elements &... __args) noexcept
- template<typename... _UTypes>
  std::tuple (_UTypes...) -> tuple< _UTypes... >
- template<typename _Alloc, typename... _UTypes>
  std::tuple (allocator_arg_t, _Alloc, _UTypes...) -> tuple< _UTypes... >
- template<typename _Alloc, typename _T1, typename _T2>
  std::tuple (allocator_arg_t, _Alloc, pair< _T1, _T2 >) -> tuple< _T1, _T2 >
- template<typename _Alloc, typename... _UTypes>
  std::tuple (allocator_arg_t, _Alloc, tuple< _UTypes... >) -> tuple< _UTypes... >
- template<typename _T1, typename _T2>
  std::tuple (pair< _T1, _T2 >) -> tuple< _T1, _T2 >

- template<typename... _Tpls, typename = typename enable_if<__and_<__is_tuple_like<_Tpls>...>::value>::type>
  constexpr auto std::tuple_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls... >::__type

Variables

- constexpr piecewise_construct_t std::piecewise_construct
- template<typename... _Types>
  constexpr size_t **std::tuple_size_v**< **const tuple**< **_Types...** > >
- template<typename... _Types>
  constexpr size_t **std::tuple_size_v**< **tuple**< **_Types...** > >
- template<typename _T1, typename _T2>
  pair (_T1, _T2) -> pair< _T1, _T2 >
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2, _U2 > > operator<=> (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)
- template<typename _T1, typename _T2>
  constexpr enable_if< __and_< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _T1, typename _T2>
  constexpr enable_if< __and_< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2, _U2 > > operator<=> (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)

3.16.1 Detailed Description

Basic function and class templates used with the rest of the library. Includes pair, swap, forward/move helpers, declval, integer_sequence.

3.16.2 Function Documentation

pair() [1/2]

```
template<typename _T1, typename _T2>
pair (
            _T1 ,
            _T2 ) -> pair< _T1, _T2 >  [related]
```

Two pairs are equal iff their members are equal.

pair() [2/2]

```
template<class _T1, class _T2>
template<typename...  _Args1, typename...  _Args2>
std::pair< _T1, _T2 >::pair (
            piecewise_construct_t ,
            tuple< _Args1...  > __first,
            tuple< _Args2...  > __second)  [inline], [constexpr]
```

"piecewise construction" using a tuple of arguments for each member.

Parameters

__first	Arguments for the first member of the pair.
__second	Arguments for the second member of the pair.

The elements of each tuple will be used as the constructor arguments for the data members of the pair.
References pair().

__addressof()

```
template<typename _Tp>
_Tp * std::__addressof (
            _Tp & __r)  [inline], [constexpr], [noexcept]
```

Same as C++11 std::addressof.
Referenced by std::back_insert_iterator< _Container >::back_insert_iterator(), std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(), std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringstream< char >::basic_stringstream(), std::basic_stringstream< char >::basic_stringstream(), std::basic_stringstream< char >::basic_stringstream(), std::front_insert_iterator< _Container >::front_insert_iterator(), std::insert_iterator< _Container >::insert_iterator(), std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator(), std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator(), std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if, std::basic_ios< _CharT, _Traits >::copyfmt(), std::forward_list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::merge(), std::sub_match< const char * >::operator<<(), __gnu_debug::_Safe_iterator< _Base_iterator, map >, __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), std::basic_string< _CharT >::operator=(), std::deque< _Tp, _Alloc >::operator=(), std::forward_list< _Tp, _Alloc >::operator=(), std::list< _Tp, _Alloc >::operator=(), std::vector< _Tp, _Alloc >::operator=(), std::sub_match< const char * >::operator==(), std::basic_istringstream< char >::rdbuf(), std::basic_ostringstream< char >::rdbuf(), std::basic_stringstream< char >::rdbuf(), std::list< _Tp, polymorphic_allocator< _Tp > >::sp, and std::list< _Tp, polymorphic_allocator< _Tp > >::splice().

__invoke()

```
template<typename _Callable, typename...  _Args>
__invoke_result< _Callable, _Args...  >::type std::__invoke (
            _Callable && __fn,
            _Args &&...  __args)  [constexpr], [noexcept]
```

Invoke a callable object.
References forward().
Referenced by std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref(), and std::once_flag::call_once.

addressof()

```
template<typename _Tp>
_Tp * std::addressof (
            _Tp & __r)  [inline], [constexpr], [noexcept]
```

Returns the actual address of the object or function referenced by r, even in the presence of an overloaded operator&.

Parameters

↩ _↩ ↩ _↩ *r*	Reference to an object or function.

Returns

The actual address.

Since

C++11

Referenced by std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref(), std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref(), _Destroy(), _Destroy_n(), std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator=(), and std::move_only_function< _Res

declval()

```
template<typename _Tp>
auto std::declval () -> decltype(__declval< _Tp >(0))  [noexcept]
```

Utility to simplify expressions used in unevaluated operands

Since

C++11

Referenced by std::vector< _Tp, polymorphic_allocator< _Tp > >::vector(), and std::unique_ptr< _State >::operator*().

forward() [1/2]

```
template<typename _Tp>
_Tp && std::forward (
            typename std::remove_reference< _Tp >::type && __t)  [constexpr], [noexcept]
```

Forward an rvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Since

C++11

forward() [2/2]

```
template<typename _Tp>
_Tp && std::forward (
            typename std::remove_reference< _Tp >::type & __t)  [constexpr], [noexcept]
```

Forward an lvalue.

Returns

The parameter cast to the specified type.

This function is used to implement "perfect forwarding".

Since

C++11

Referenced by std::experimental::fundamentals_v1::any::any(), std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexc
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function(), std::copyable_function< _R
std::function< _Res(_ArgTypes...)>::function(), std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::functio
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function(),
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function(),
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function(),
std::pair< const __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator > >::pair(), std::unique_ptr< _State >::unique_ptr(),
__invoke(), _Construct(), std::list< _Tp, polymorphic_allocator< _Tp > >::_M_create_node(), std::deque< _Tp, _Alloc >::_M_push_bac
std::deque< _Tp, _Alloc >::_M_push_front_aux(), std::shared_ptr< _State_base >::allocate_shared(), async(),
async(), bind(), std::once_flag::call_once, std::allocator_traits< _OuterAlloc >::construct(), std::allocator_traits< allocator< _Tp > >::cor
std::allocator_traits< allocator< void > >::construct(), std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::construct(),
std::deque< _Tp, _Alloc >::emplace(), std::list< _Tp, _Alloc >::emplace(), std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< c
std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace(), std::multiset< _Key, _Cmp, polymo
std::set< _Key, _Cmp, polymorphic_allocator< _Key > >::emplace(), std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allo
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp
> >::emplace(), std::unordered_multiset< _Key, _Hash, _Pred, polymorphic_allocator< _Key > >::emplace(),
std::unordered_set< _Key, _Hash, _Pred, polymorphic_allocator< _Key > >::emplace(), std::vector< _Tp, polymorphic_allocator< _Tp
std::forward_list< _Tp, polymorphic_allocator< _Tp > >::emplace_after(), std::forward_list< _Tp, polymorphic_allocator< _Tp > >::em
std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace_hint(), std::multimap< _Key, _Tp, _Cmp, p
std::multiset< _Key, _Cmp, polymorphic_allocator< _Key > >::emplace_hint(), std::set< _Key, _Cmp, polymorphic_allocator< _Key >
std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace↩
_hint(), std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > >
>::emplace_hint(), std::unordered_multiset< _Key, _Hash, _Pred, polymorphic_allocator< _Key > >::emplace_hint(),
std::unordered_set< _Key, _Hash, _Pred, polymorphic_allocator< _Key > >::emplace_hint(), std::pair< const __iter_key_t< _InputItera
std::pair< const __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator > >::explicit(), std::pair< const __iter_key_t< _InputIterato
forward_as_tuple(), get(), get(), get(), get(), get(), get(), inout_ptr(), std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Ke
std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::multimap< _Key, _Tp, _Cmp, polymor
std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::unordered_map< _Key, _Tp, _Ha
std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(),
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(),
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(),
std::unordered_set< _Key, _Hash, _Pred, polymorphic_allocator< _Key > >::insert(), std::unordered_set< _Key, _Hash, _Pred, polymo
std::experimental::make_array(), std::experimental::make_ostream_joiner(), make_pair(), std::shared_ptr< _State_base >::make_shared
make_tuple(), std::experimental::not_fn(), std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operato
std::function< _Res(_ArgTypes...)>::operator()(), std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::opera
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator()(), std::copyable_function< _Res(_Ar
std::experimental::fundamentals_v1::any::operator=(), std::function< _Res(_ArgTypes...)>::operator=(), std::move_only_function< _Res(
std::pair< const __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator > >::operator=(), std::pair< const __iter_key_t< _InputItera
std::pair< const __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator > >::operator=(), std::pair< const __iter_key_t< _InputItera
std::unique_ptr< _State >::operator=(), std::unique_ptr< _Tp[], _Dp >::operator=(), operator>>(), std::unordered_map< _Key, _Tp, _H
out_ptr(), std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > >::reserve(), std::experimental::sample(),
sample(), throw_with_nested(), and tuple_cat().

forward_as_tuple()

```
template<typename...  _Elements>
tuple< _Elements &&...  > std::forward_as_tuple (
            _Elements &&...  __args)  [constexpr], [noexcept]
```

Create a tuple of lvalue or rvalue references to the arguments.
References forward().

forward_like()

```
template<typename _Tp, typename _Up>
__like_t< _Tp, _Up > std::forward_like (
            _Up && __x)  [nodiscard], [constexpr], [noexcept]
```

Forward with the cv-qualifiers and value category of another type.

Template Parameters

_Tp	An lvalue reference or rvalue reference.
_Up	An lvalue reference type deduced from the function argument.

Parameters

_↩ _x	An lvalue.

Returns

__x converted to match the qualifiers of _Tp.

Since

C++23

get() [1/4]

```
template<size_t __i, typename...  _Elements>
const __tuple_element_t< __i, tuple< _Elements...  > > && std::get (
            const tuple< _Elements...  > && __t)  [constexpr], [noexcept]
```

Return a const rvalue reference to the ith element of a const tuple rvalue.
References forward().

get() [2/4]

```
template<size_t __i, typename...  _Elements>
const __tuple_element_t< __i, tuple< _Elements...  > > & std::get (
            const tuple< _Elements...  > & __t)  [constexpr], [noexcept]
```

Return a const reference to the ith element of a const tuple.

get() [3/4]

```
template<size_t __i, typename...  _Elements>
__tuple_element_t< __i, tuple< _Elements...  > > && std::get (
            tuple< _Elements...  > && __t)  [constexpr], [noexcept]
```

Return an rvalue reference to the ith element of a tuple rvalue.
References forward().

get() [4/4]

```
template<size_t __i, typename... _Elements>
__tuple_element_t< __i, tuple< _Elements... > > & std::get (
            tuple< _Elements... > & __t) [constexpr], [noexcept]
```

Return a reference to the ith element of a tuple.

make_pair()

```
template<typename _T1, typename _T2>
pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type >
std::make_pair (
            _T1 && __x,
            _T2 && __y) [constexpr]
```

A convenience wrapper for creating a pair from two objects.

Parameters

_↩ _x	The first object.
_↩ _y	The second object.

Returns

A newly-constructed pair<> object of the appropriate type.

The C++98 standard says the objects are passed by reference-to-const, but C++03 says they are passed by value (this was LWG issue #181).
Since C++11 they have been passed by forwarding reference and then forwarded to the new members of the pair. To create a pair with a member of reference type, pass a `reference_wrapper` to this function.
References forward().
Referenced by __gnu_parallel::__find_template(), __gen_two_uniform_ints(), __gnu_debug::__get_distance(), __gnu_parallel::__parallel_merge_advance(), __gnu_parallel::__parallel_sort_qsb(), __gnu_parallel::__qsb_local_sort_with_helping(), __gnu_parallel::__adjacent_find_selector::_M_sequential_algorithm(), __gnu_parallel::__find_first_of_selector< _FIterator >::_M_seque __gnu_parallel::__find_if_selector::_M_sequential_algorithm(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_parallel::parallel_multiway_merge(), __gnu_parallel::parallel_sort_mwms_pu(), and __gnu_pbds::detail::pat_↩ trie_base::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >::valid_prefix().

make_tuple()

```
template<typename... _Elements>
tuple< typename __decay_and_strip< _Elements >::__type... > std::make_tuple (
            _Elements &&... __args) [constexpr]
```

Create a tuple containing copies of the arguments.
References forward().

move()

```
template<typename _Tp>
std::remove_reference< _Tp >::type && std::move (
            _Tp && __t) [constexpr], [noexcept]
```

Convert a value to an rvalue.

Parameters

↩ _↩ ↩ _↩ *t*	A thing of arbitrary type.

Returns

The parameter cast to an rvalue-reference to allow moving it.

Since

C++11

Referenced by __gnu_cxx::__versa_string< char >::__versa_string(), std::basic_string< _CharT >::basic_string(), std::deque< _Tp, polymorphic_allocator< _Tp > >::deque(), std::discard_block_engine< ranlux24_base, 223, 23 >::discard_block_eng std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset(), std::forward_list< _Tp, polymorphic_allocator< _Tp > >::forward_list(), std::future< _Res >::future(), std::future< _Res & >::future(), std::future< void >::future(), std::independent_bits_engine< _RandomNu std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::map(), std::match_results< ↩ _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::↩ match_results(), std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::multimap(), std::multiset< _Key, _Cmp, polymorphic_allocator< _Key > >::multiset(), std::priority_queue< _Tp, _Sequence, _Compare >::priority_q std::set< _Key, _Cmp, polymorphic_allocator< _Key > >::set(), std::shared_future< _Res >::shared_future(), std::shared_future< _Res >::shared_future(), std::shared_future< _Res & >::shared_future(), std::shared_future< _Res & >::shared_fu std::shared_future< void >::shared_future(), std::shared_future< void >::shared_future(), std::shared_ptr< _State_base >::shared_ptr(std::shared_ptr< _State_base >::shared_ptr(), std::shared_ptr< _State_base >::shared_ptr(), std::shared_ptr< _State_base >::shared std::shared_ptr< _State_base >::shared_ptr(), std::shared_ptr< _State_base >::shared_ptr(), std::shared_ptr< _State_base >::shared std::shuffle_order_engine< minstd_rand0, 256 >::shuffle_order_engine(), std::unique_ptr< _State >::unique_ptr(), std::unique_ptr< _Tp[], _Dp >::unique_ptr(), std::vector< _Tp, polymorphic_allocator< _Tp > >::vector(), std::unique_ptr< _State >::~ std::experimental::any_cast(), std::basic_regex< char >::assign(), std::basic_string< _CharT >::assign(), async(), atomic_compare_exchange_strong(), atomic_compare_exchange_strong(), atomic_compare_exchange_strong_explicit(), atomic_compare_exchange_strong_explicit(), atomic_compare_exchange_weak(), atomic_compare_exchange_weak(), atomic_compare_exchange_weak_explicit(), atomic_compare_exchange_weak_explicit(), atomic_exchange(), atomic_exchange(), atomic_store(), atomic_store(), std::shared_ptr< _State_base >::const_pointer_cast(), copy_n(), distance(), std::shared_ptr< _State_base >::dynamic_pointer_cast(), exclusive_scan(), exclusive_scan(), for_each_n(), std::future< _Res >::get(), get(), get(), inclusive_scan(), std::deque< _Tp, polymorphic_allocator< _Tp > >::insert(), std::list< _Tp, polymorphic_allocator< _Tp > >::insert(), std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::multimap< _Key, _Tp, _Cmp, polymor std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::set< _Key, _Cmp, polymorphic_a std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::unordered_multimap< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), std::unordered_multiset< _Key, _Hash, _Pred, polymorphic_allocator< _Key > >::insert(), std::unordered_multiset< _Key, _Hash, _Pre std::unordered_set< _Key, _Hash, _Pred, polymorphic_allocator< _Key > >::insert(), std::unordered_set< _Key, _Hash, _Pred, polymo std::vector< _Tp, _Alloc >::insert(), std::vector< _Tp, polymorphic_allocator< _Tp > >::insert(), std::experimental::make_boyer_moore_ std::experimental::make_boyer_moore_searcher(), std::forward_list< _Tp, _Alloc >::merge(), std::forward_list< _Tp, polymorphic_alloca std::forward_list< _Tp, polymorphic_allocator< _Tp > >::merge(), std::list< _Tp, polymorphic_allocator< _Tp > >::merge(), std::list< _Tp, polymorphic_allocator< _Tp > >::merge(), move_if_noexcept(), std::tr2::dynamic_bitset< _WordT, _Alloc >::operator&=(std::chrono::operator<<(), std::deque< _Tp, polymorphic_allocator< _Tp > >::operator=(), std::forward_list< _Tp, polymorphic_allocat std::function< _Res(_ArgTypes...)>::operator=(), std::list< _Tp, polymorphic_allocator< _Tp > >::operator=(), std::vector< _Tp, polymorphic_allocator< _Tp > >::operator=(), std::filesystem::path::operator>>, operator>>(), std::unordered_map< _Key, _Tp, _Hash, _Pred, polymorphic_allocator< pair< const _Key, _Tp > > >::operator[](),

reduce(), std::shared_ptr< _State_base >::reinterpret_pointer_cast(), std::unique_ptr< _State >::reset(), std::unique_ptr< _Tp[], _Dp >
std::list< _Tp, polymorphic_allocator< _Tp > >::splice(), std::list< _Tp, polymorphic_allocator< _Tp > >::splice(),
std::forward_list< _Tp, polymorphic_allocator< _Tp > >::splice_after(), std::shared_ptr< _State_base >::static_pointer_cast(),
transform_exclusive_scan(), transform_inclusive_scan(), and transform_reduce().

move_if_noexcept()

```
template<typename _Tp>
__conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > std::move_if_↩
noexcept (
            _Tp & __x)  [constexpr], [noexcept]
```

Conditionally convert a value to an rvalue.

Parameters

_↩ _x	A thing of arbitrary type.

Returns

The parameter, possibly cast to an rvalue-reference.

Same as std::move unless the type's move constructor could throw and the type is copyable, in which case an lvalue-reference is returned instead.

Since

C++11

References move().

operator<=>() [1/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2,
_U2 > > operator<=> (
            const pair< _T1, _T2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Defines a lexicographical order for pairs.
For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

operator<=>() [2/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2,
_U2 > > operator<=> (
            const pair< _T1, _T2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Defines a lexicographical order for pairs.
For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

operator==() [1/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
bool operator== (
            const pair< _T1, _T2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Two pairs are equal iff their members are equal.

operator==() [2/2]

```
template<typename _T1, typename _T2, typename _U1, typename _U2>
bool operator== (
            const pair< _T1, _T2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Two pairs are equal iff their members are equal.

swap() [1/5]

```
template<typename _T1, typename _T2>
enable_if< __and_< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (
            pair< _T1, _T2 > & __x,
            pair< _T1, _T2 > & __y)  [related]
```

Swap overload for pairs. Calls std::pair::swap().

Note

This std::swap overload is not declared in C++03 mode, which has performance implications, e.g. see https://gcc.gnu.org/PR38466

swap() [2/5]

```
template<typename _T1, typename _T2>
enable_if< __and_< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (
            pair< _T1, _T2 > & __x,
            pair< _T1, _T2 > & __y)  [related]
```

Swap overload for pairs. Calls std::pair::swap().

Note

This std::swap overload is not declared in C++03 mode, which has performance implications, e.g. see https://gcc.gnu.org/PR38466

swap() [3/5]

```
template<typename _Tp>
void std::swap (
            _Tp & __a,
            _Tp & __b)  [inline], [constexpr], [noexcept]
```

Swaps two values.

Parameters

_↩ _a	A thing of arbitrary type.

_↩ _b	Another thing of arbitrary type.

swap() [4/5]

```
template<typename _Tp, size_t _Nm>
void std::swap (
            _Tp(&) __a[_Nm],
            _Tp(&) __b[_Nm])  [inline], [constexpr], [noexcept]
```

Swap the contents of two arrays.

swap() [5/5]

```
template<typename...  _Elements>
enable_if< __and_< __is_swappable< _Elements >...  >::value >::type std::swap (
            tuple< _Elements...  > & __x,
            tuple< _Elements...  > & __y)  [inline], [constexpr], [delete], [noexcept]
```

Exchange the values of two tuples.

tie()

```
template<typename...  _Elements>
tuple< _Elements &...  > std::tie (
            _Elements &...  __args)  [constexpr], [noexcept]
```

Return a tuple of lvalue references bound to the arguments.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt().

tuple_cat()

```
template<typename...  _Tpls, typename = typename enable_if<__and_<__is_tuple_like<_Tpls>...>::↩
value>::type>
auto std::tuple_cat (
            _Tpls &&...  __tpls) -> typename __tuple_cat_result< _Tpls...  >::__type  [constexpr]
```

Create a `tuple` containing all elements from multiple tuple-like objects.
References forward().

3.16.3 Variable Documentation

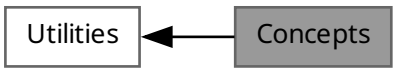
piecewise_construct

```
piecewise_construct_t std::piecewise_construct  [inline], [constexpr]
```

Tag for piecewise construction of std::pair objects.
Referenced by std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::operator[]().

3.16.4 Concepts

Collaboration diagram for Concepts:



Files

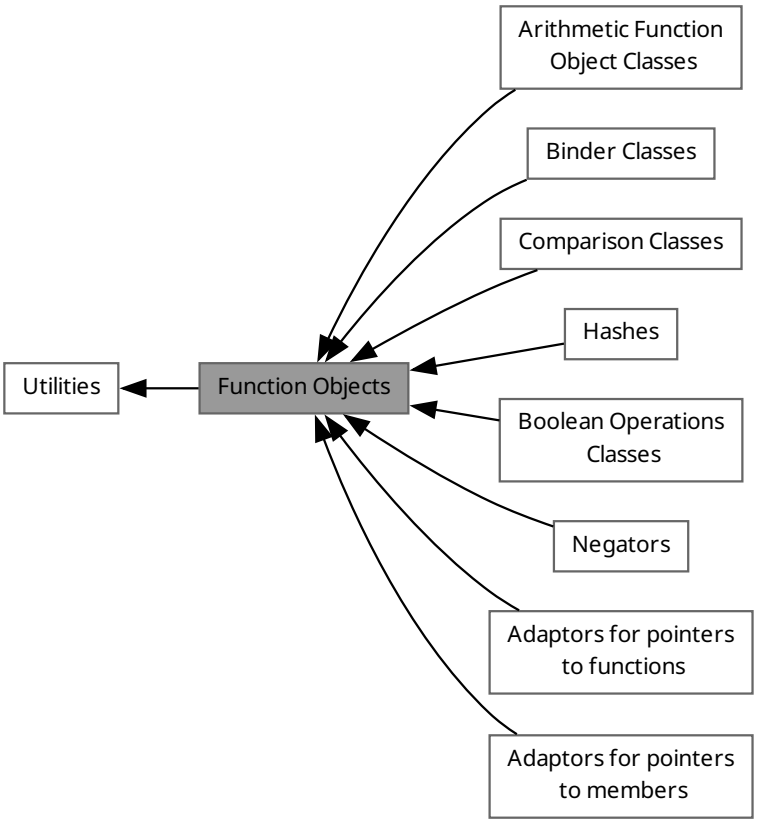
- file concepts
- file ranges

3.16.4.1 Detailed Description

Concepts for checking type requirements.

3.16.5 Function Objects

Collaboration diagram for Function Objects:



Topics

- Adaptors for pointers to functions
- Adaptors for pointers to members
- Arithmetic Function Object Classes
- Binder Classes
- Boolean Operations Classes
- Comparison Classes
- Hashes
- Negators

Classes

- struct std::binary_function< _Arg1, _Arg2, _Result >
- class std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>
- class std::function< _Res(_ArgTypes...)>
- class std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

- class std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>
- class std::reference_wrapper< _Tp >
- struct std::unary_function< _Arg, _Result >

Functions

- template<typename _Tp, typename _Class>
  constexpr _Mem_fn< _Tp _Class::* > std::mem_fn (_Tp _Class::*__pm) noexcept

3.16.5.1 Detailed Description

Function objects, or *functors*, are objects with an `operator()` defined and accessible. They can be passed as arguments to algorithm templates and used in place of a function pointer. Not only is the resulting expressiveness of the library increased, but the generated code can be more efficient than what you might write by hand. When we refer to *functors*, then, generally we include function pointers in the description as well.
Often, functors are only created as temporaries passed to algorithm calls, rather than being created as named variables. Two examples taken from the standard itself follow. To perform a by-element addition of two vectors `a` and `b` containing `double`, and put the result in `a`, use

```
transform (a.begin(), a.end(), b.begin(), a.begin(), plus<double>());
```

To negate every element in `a`, use

```
transform(a.begin(), a.end(), a.begin(), negate<double>());
```

The addition and negation functions will usually be inlined directly.
An *adaptable function object* is one which provides nested typedefs `result_type` and either `argument_type` (for a unary function) or `first_argument_type` and `second_argument_type` (for a binary function). Those typedefs are used by function object adaptors such as `bind2nd`. The standard library provides two class templates, `unary_function` and `binary_function`, which define those typedefs and so can be used as base classes of adaptable function objects.
Since C++11 the use of function object adaptors has been superseded by more powerful tools such as lambda expressions, `function<>`, and more powerful type deduction (using `auto` and `decltype`). The helpers for defining adaptable function objects are deprecated since C++11, and no longer part of the standard library since C++17. However, they are still defined and used by libstdc++ after C++17, as a conforming extension.

3.16.5.2 Function Documentation

mem_fn()

```
template<typename _Tp, typename _Class>
_Mem_fn< _Tp _Class::* > std::mem_fn (
            _Tp _Class::* __pm)  [inline], [constexpr], [noexcept]
```

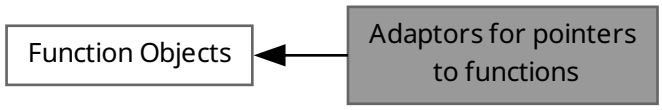
Returns a function object that forwards to the member pointer pointer `pm`.
This allows a pointer-to-member to be transformed into a function object that can be called with an object expression as its first argument.
For a pointer-to-data-member the result must be called with exactly one argument, the object expression that would be used as the first operand in a `obj.*memptr` or `objp->*memptr` expression.
For a pointer-to-member-function the result must be called with an object expression and any additional arguments to pass to the member function, as in an expression like `(obj.*memfun)(args...)` or `(objp->*memfun)(args...)`.
The object expression can be a pointer, reference, `reference_wrapper`, or smart pointer, and the call wrapper will dereference it as needed to apply the pointer-to-member.

Since

C++11

3.16.5.3 Adaptors for pointers to functions

Collaboration diagram for Adaptors for pointers to functions:



Classes

- class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >
- class std::pointer_to_unary_function< _Arg, _Result >

Functions

- template<typename _Arg, typename _Result>
  pointer_to_unary_function< _Arg, _Result > std::ptr_fun (_Result(*__x)(_Arg))
- template<typename _Arg1, typename _Arg2, typename _Result>
  pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (_Result(*__x)(_Arg1, _Arg2))

3.16.5.3.1 Detailed Description

The advantage of function objects over pointers to functions is that the objects in the standard library declare nested typedefs describing their argument and result types with uniform names (e.g., `result_type` from the base classes `unary_function` and `binary_function`). Sometimes those typedefs are required, not just optional.
Adaptors are provided to turn pointers to unary (single-argument) and binary (double-argument) functions into function objects. The long-winded functor `pointer_to_unary_function` is constructed with a function pointer `f`, and its `operator()` called with argument `x` returns `f(x)`. The functor `pointer_to_binary_function` does the same thing, but with a double-argument `f` and `operator()`.
The function `ptr_fun` takes a pointer-to-function `f` and constructs an instance of the appropriate functor.

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

3.16.5.3.2 Function Documentation

ptr_fun() [1/2]

```
template<typename _Arg, typename _Result>
pointer_to_unary_function< _Arg, _Result > std::ptr_fun (
            _Result(* __x )(_Arg))  [inline]
```

One of the adaptors for function pointers.
References ptr_fun().
Referenced by ptr_fun(), and ptr_fun().

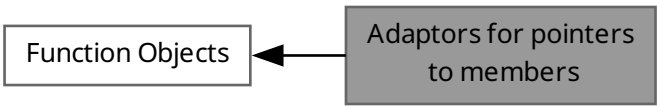
ptr_fun() [2/2]

```
template<typename _Arg1, typename _Arg2, typename _Result>
pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (
            _Result(* __x )(_Arg1, _Arg2))  [inline]
```

One of the adaptors for function pointers.
References ptr_fun().

3.16.5.4 Adaptors for pointers to members

Collaboration diagram for Adaptors for pointers to members:



Classes

- class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >
- class std::const_mem_fun1_t< _Ret, _Tp, _Arg >
- class std::const_mem_fun_ref_t< _Ret, _Tp >
- class std::const_mem_fun_t< _Ret, _Tp >
- class std::mem_fun1_ref_t< _Ret, _Tp, _Arg >
- class std::mem_fun1_t< _Ret, _Tp, _Arg >
- class std::mem_fun_ref_t< _Ret, _Tp >
- class std::mem_fun_t< _Ret, _Tp >

Functions

- template<typename _Ret, typename _Tp>
  mem_fun_t< _Ret, _Tp > **std::mem_fun** (_Ret(_Tp::*__f)())
- template<typename _Ret, typename _Tp, typename _Arg>
  mem_fun1_t< _Ret, _Tp, _Arg > **std::mem_fun** (_Ret(_Tp::*__f)(_Arg))
- template<typename _Ret, typename _Tp>
  mem_fun_ref_t< _Ret, _Tp > **std::mem_fun_ref** (_Ret(_Tp::*__f)())
- template<typename _Ret, typename _Tp, typename _Arg>
  mem_fun1_ref_t< _Ret, _Tp, _Arg > **std::mem_fun_ref** (_Ret(_Tp::*__f)(_Arg))

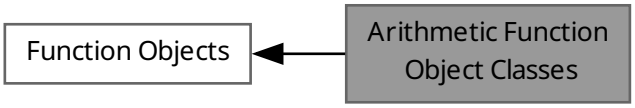
3.16.5.4.1 Detailed Description

There are a total of 8 = 2^3 function objects in this family. (1) Member functions taking no arguments vs member functions taking one argument. (2) Call through pointer vs call through reference. (3) Const vs non-const member function.
All of this complexity is in the function objects themselves. You can ignore it by using the helper function `mem_fun` and `mem_fun_ref`, which create whichever type of adaptor is appropriate.

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

3.16.5.5 Arithmetic Function Object Classes

Collaboration diagram for Arithmetic Function Object Classes:



Classes

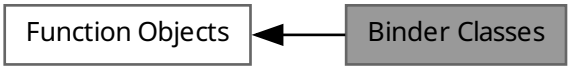
- struct std::divides< _Tp >
- struct std::divides< void >
- struct std::minus< _Tp >
- struct std::minus< void >
- struct std::modulus< _Tp >
- struct std::modulus< void >
- struct std::multiplies< _Tp >
- struct std::multiplies< void >
- struct std::negate< _Tp >
- struct std::negate< void >
- struct std::plus< _Tp >

3.16.5.5.1 Detailed Description

The library provides function objects for basic arithmetic operations. See the documentation for function objects for examples of their use.

3.16.5.6 Binder Classes

Collaboration diagram for Binder Classes:



Namespaces

- namespace std::placeholders

Classes

- struct std::_Placeholder< _Num >
- class std::binder1st< _Operation >
- class std::binder2nd< _Operation >
- struct std::is_bind_expression< _Tp >
- struct std::is_bind_expression< _Bind< _Signature > >
- struct std::is_bind_expression< _Bind_result< _Result, _Signature > >
- struct std::is_bind_expression< const _Bind< _Signature > >
- struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >
- struct std::is_bind_expression< const volatile _Bind< _Signature > >
- struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >
- struct std::is_bind_expression< volatile _Bind< _Signature > >
- struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >
- struct std::is_placeholder< _Tp >
- struct std::is_placeholder< _Placeholder< _Num > >

Functions

- template<typename _Result, typename _Func, typename... _BoundArgs>
  constexpr _Bindres_helper< _Result, _Func, _BoundArgs... >::type std::bind (_Func &&__f, _BoundArgs &&... __args)
- template<typename _Func, typename... _BoundArgs>
  constexpr _Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type std::bind (_Func &&↩ __f, _BoundArgs &&... __args)
- template<typename _Operation, typename _Tp>
  binder1st< _Operation > std::bind1st (const _Operation &__fn, const _Tp &__x)
- template<typename _Operation, typename _Tp>
  binder2nd< _Operation > std::bind2nd (const _Operation &__fn, const _Tp &__x)

3.16.5.6.1 Detailed Description

Binders turn functions/functors with two arguments into functors with a single argument, storing an argument to be applied later. For example, a variable `B` of type `binder1st` is constructed from a functor `f` and an argument `x`. Later, B's `operator()` is called with a single argument `y`. The return value is the value of `f(x,y)`. `B` can be *called* with various arguments (y1, y2, ...) and will in turn call `f(x,y1)`, `f(x,y2)`, ...

The function `bind1st` is provided to save some typing. It takes the function and an argument as parameters, and returns an instance of `binder1st`.

The type `binder2nd` and its creator function `bind2nd` do the same thing, but the stored argument is passed as the second parameter instead of the first, e.g., `bind2nd`(std::minus<float>(),1.3) will create a functor whose `operator()` accepts a floating-point number, subtracts 1.3 from it, and returns the result. (If `bind1st` had been used, the functor would perform *1.3 - x* instead.

Creator-wrapper functions like `bind1st` are intended to be used in calling algorithms. Their return values will be temporary objects. (The goal is to not require you to type names like `std::binder1st<std::plus<int>>` for declaring a variable to hold the return value from `bind1st(std::plus<int>(),5)`.

These become more useful when combined with the composition functions.

These functions are deprecated in C++11 and can be replaced by `std::bind` (or `std::tr1::bind`) which is more powerful and flexible, supporting functions with any number of arguments. Uses of `bind1st` can be replaced by `std::bind(f, x, std::placeholders::_1)` and `bind2nd` by `std::bind(f, std::↩ placeholders::_1, x)`.

3.16.5.6.2 Function Documentation

bind() [1/2]

```
template<typename _Result, typename _Func, typename...  _BoundArgs>
_Bindres_helper< _Result, _Func, _BoundArgs...  >::type std::bind (
            _Func && __f,
            _BoundArgs &&...  __args)  [inline], [constexpr]
```

Function template for std::bind<R>.

Since

C++11

References forward().

bind() [2/2]

```
template<typename _Func, typename...  _BoundArgs>
_Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs...  >::type std::bind (
            _Func && __f,
            _BoundArgs &&...  __args)  [inline], [constexpr]
```

Function template for std::bind.

Since

C++11

References forward().

bind1st()

```
template<typename _Operation, typename _Tp>
binder1st< _Operation > std::bind1st (
            const _Operation & __fn,
            const _Tp & __x)  [inline]
```

One of the binder functors.
References bind1st().
Referenced by bind1st().

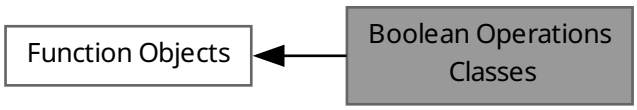
bind2nd()

```
template<typename _Operation, typename _Tp>
binder2nd< _Operation > std::bind2nd (
            const _Operation & __fn,
            const _Tp & __x)  [inline]
```

One of the binder functors.
References bind2nd().
Referenced by bind2nd().

3.16.5.7 Boolean Operations Classes

Collaboration diagram for Boolean Operations Classes:



Classes

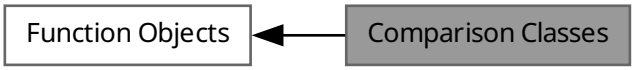
- struct std::logical_and< _Tp >
- struct std::logical_and< void >
- struct std::logical_not< _Tp >
- struct std::logical_not< void >
- struct std::logical_or< _Tp >
- struct std::logical_or< void >

3.16.5.7.1 Detailed Description

The library provides function objects for the logical operations: `&&`, ||, and `!`.

3.16.5.8 Comparison Classes

Collaboration diagram for Comparison Classes:



Classes

- struct std::equal_to< _Tp >
- struct std::equal_to< void >
- struct std::greater< _Tp >
- struct std::greater< void >
- struct std::greater_equal< _Tp >
- struct std::greater_equal< void >
- struct std::less< _Tp >
- struct std::less< void >
- struct std::less_equal< _Tp >

- struct std::less_equal< void >
- struct std::not_equal_to< _Tp >
- struct std::not_equal_to< void >

3.16.5.8.1 Detailed Description

The library provides six wrapper functors for all the basic comparisons in C++, like <.

3.16.5.9 Hashes

Collaboration diagram for Hashes:



Classes

- struct std::hash< _Tp >
- struct std::hash< _Tp * >
- struct std::hash< bool >
- struct std::hash< char >
- struct std::hash< char16_t >
- struct std::hash< char32_t >
- struct std::hash< double >
- struct std::hash< float >
- struct std::hash< int >
- struct std::hash< long >
- struct std::hash< long double >
- struct std::hash< long long >
- struct std::hash< short >
- struct std::hash< signed char >
- struct std::hash< unsigned char >
- struct std::hash< unsigned int >
- struct std::hash< unsigned long >
- struct std::hash< unsigned long long >
- struct std::hash< unsigned short >
- struct std::hash< wchar_t >

Macros

- #define **_Cxx_hashtable_define_trivial_hash**(_Tp)

Variables

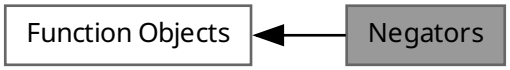
- template<typename _Tp, typename = void>
  constexpr bool **std::__is_hash_enabled_for**
- template<typename _Tp>
  constexpr bool **std::__is_hash_enabled_for**< **_Tp, __void_t**< **decltype(hash**< **_Tp** >()(**declval**< **_Tp** >()))> >

3.16.5.9.1 Detailed Description

Hashing functors taking a variable type and returning a `std::size_t`.

3.16.5.10 Negators

Collaboration diagram for Negators:



Classes

- class std::binary_negate< _Predicate >
- class std::unary_negate< _Predicate >

Functions

- template<typename _Predicate>
  constexpr unary_negate< _Predicate > std::not1 (const _Predicate &__pred)
- template<typename _Predicate>
  constexpr binary_negate< _Predicate > std::not2 (const _Predicate &__pred)

3.16.5.10.1 Detailed Description

The function templates `not1` and `not2` are function object adaptors, which each take a predicate functor and wrap it in an instance of `unary_negate` or `binary_negate`, respectively. Those classes are functors whose `operator()` evaluates the wrapped predicate function and then returns the negation of the result.
For example, given a vector of integers and a trivial predicate,

```
struct IntGreaterThanThree
  : public std::unary_function<int, bool>
{
    bool operator() (int x) const { return x > 3; }
};

std::find_if (v.begin(), v.end(), not1(IntGreaterThanThree()));
```

The call to `find_if` will locate the first index (i) of `v` for which `!(v[i] > 3)` is true.
The not1/unary_negate combination works on predicates taking a single argument. The not2/binary_negate combination works on predicates taking two arguments.

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

3.16.5.10.2 Function Documentation

not1()

```
template<typename _Predicate>
unary_negate< _Predicate > std::not1 (
            const _Predicate & __pred)  [inline], [constexpr]
```

One of the negation functors.
References not1().
Referenced by not1().

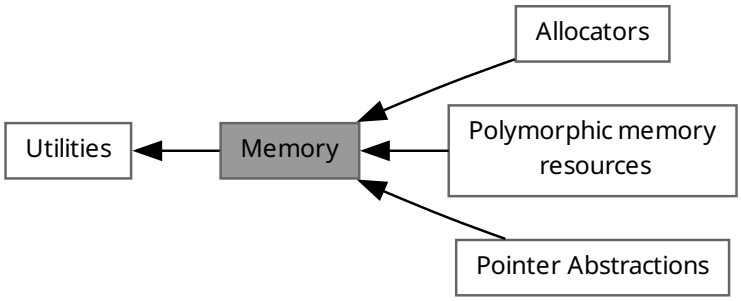
not2()

```
template<typename _Predicate>
binary_negate< _Predicate > std::not2 (
            const _Predicate & __pred)  [inline], [constexpr]
```

One of the negation functors.
References not2().
Referenced by not2().

3.16.6 Memory

Collaboration diagram for Memory:



Topics

- Allocators
- Pointer Abstractions
- Polymorphic memory resources

Files

- file memory

Functions

- void * std::align (size_t __align, size_t __size, void *&__ptr, size_t &__space) noexcept

- template<size_t _Align, class _Tp>
  constexpr _Tp * std::assume_aligned (_Tp *__ptr) noexcept
- template<size_t _Align, class _Tp>
  bool std::is_sufficiently_aligned (_Tp *__ptr)
- template<typename _InputIterator, typename _ForwardIterator>
  constexpr _ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator>
  constexpr _ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)
- template<typename _ForwardIterator>
  constexpr void std::uninitialized_default_construct (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator std::uninitialized_default_construct_n (_ForwardIterator __first, _Size __count)
- template<typename _ForwardIterator, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>
  constexpr void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)
- template<typename _ForwardIterator, typename _Size, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>
  constexpr _ForwardIterator std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)
- template<typename _InputIterator, typename _ForwardIterator>
  constexpr _ForwardIterator std::uninitialized_move (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator>
  constexpr pair< _InputIterator, _ForwardIterator > std::uninitialized_move_n (_InputIterator __first, _Size __count, _ForwardIterator __result)
- template<typename _ForwardIterator>
  constexpr void std::uninitialized_value_construct (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator std::uninitialized_value_construct_n (_ForwardIterator __first, _Size __count)

3.16.6.1 Detailed Description

Components for memory allocation, deallocation, and management.

3.16.6.2 Function Documentation

align()

```
void * std::align (
            size_t __align,
            size_t __size,
            void *& __ptr,
            size_t & __space)  [inline], [noexcept]
```

Fit aligned storage in buffer.
This function tries to fit *__size* bytes of storage with alignment *__align* into the buffer *__ptr* of size *__space* bytes. If such a buffer fits then *__ptr* is changed to point to the first byte of the aligned storage and *__space* is reduced by the bytes used for alignment.
C++11 20.6.5 [ptr.align]

Parameters

__align	A fundamental or extended alignment value.
__size	Size of the aligned storage required.
__ptr	Pointer to a buffer of *__space* bytes.

__space	Size of the buffer pointed to by *__ptr*.

Returns

the updated pointer if the aligned storage fits, otherwise nullptr.

assume_aligned()

```
template<size_t _Align, class _Tp>
_Tp * std::assume_aligned (
            _Tp * __ptr)  [nodiscard], [constexpr], [noexcept]
```

Inform the compiler that a pointer is aligned.

Template Parameters

_Align	An alignment value (i.e. a power of two)
_Tp	An object type

Parameters

__ptr	A pointer that is aligned to _Align

C++20 20.10.6 [ptr.align]

is_sufficiently_aligned()

```
template<size_t _Align, class _Tp>
bool std::is_sufficiently_aligned (
            _Tp * __ptr)  [inline], [nodiscard]
```

Is `__ptr` aligned to an _Align byte boundary?

Template Parameters

_Align	An alignment value
_Tp	An object type

C++26 20.2.5 [ptr.align]

uninitialized_copy()

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

Copies the range [first,last) into result.

Parameters

__first	An input iterator.
__last	An input iterator.
__result	A forward iterator.

Returns

__result + (__last - __first)

Like std::copy, but does not require an initialized output range.
References to_address().
Referenced by __gnu_parallel::parallel_sort_mwms_pu(), and uninitialized_move().

uninitialized_copy_n()

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

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

Parameters

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

Returns

__result + __n

Since

C++11

Like copy_n(), but does not require an initialized output range.

uninitialized_default_construct()

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

Default-initializes objects in the range [first,last).

Parameters

__first	A forward iterator.

__last	A forward iterator.

Since

C++17

uninitialized_default_construct_n()

```
template<typename _ForwardIterator, typename _Size>
_ForwardIterator std::uninitialized_default_construct_n (
            _ForwardIterator __first,
            _Size __count)  [inline], [constexpr]
```

Default-initializes objects in the range [first,first+count).

Parameters

__first	A forward iterator.
__count	The number of objects to construct.

Returns

__first + __count

Since

C++17

uninitialized_fill()

```
template<typename _ForwardIterator, typename _Tp = typename iterator_traits<_ForwardIterator>::↩
value_type>
void std::uninitialized_fill (
            _ForwardIterator __first,
            _ForwardIterator __last,
            const _Tp & __x)  [inline], [constexpr]
```

Copies the value x into the range [first,last).

Parameters

__first	A forward iterator.
__last	A forward iterator.
__x	The source value.

Like std::fill, but does not require an initialized output range.
References to_address().

uninitialized_fill_n()

```
template<typename _ForwardIterator, typename _Size, typename _Tp = typename iterator_traits<_↩
ForwardIterator>::value_type>
```

```
_ForwardIterator std::uninitialized_fill_n (
            _ForwardIterator __first,
            _Size __n,
            const _Tp & __x)  [inline], [constexpr]
```

Copies the value x into the range [first,first+n).

Parameters

__first	A forward iterator.
__n	The number of copies to make.
__x	The source value.

Returns

__first + __n.

Like std::fill_n, but does not require an initialized output range.
References to_address().

uninitialized_move()

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

Move-construct from the range [first,last) into result.

Parameters

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

Returns

__result + (__first - __last)

Since

C++17

References uninitialized_copy().

uninitialized_move_n()

```
template<typename _InputIterator, typename _Size, typename _ForwardIterator>
pair< _InputIterator, _ForwardIterator > std::uninitialized_move_n (
            _InputIterator __first,
            _Size __count,
            _ForwardIterator __result)  [inline], [constexpr]
```

Move-construct from the range [first,first+count) into result.

Parameters

__first	An input iterator.
__count	The number of objects to initialize.
__result	An output iterator.

Returns

__result + __count

Since

C++17

uninitialized_value_construct()

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

Value-initializes objects in the range [first,last).

Parameters

__first	A forward iterator.
__last	A forward iterator.

Since

C++17

uninitialized_value_construct_n()

```
template<typename _ForwardIterator, typename _Size>
_ForwardIterator std::uninitialized_value_construct_n (
            _ForwardIterator __first,
            _Size __count)  [inline], [constexpr]
```

Value-initializes objects in the range [first,first+count).

Parameters

__first	A forward iterator.
__count	The number of objects to construct.

Returns

__result + __count

Since

C++17

3.16.6.3 Allocators

Collaboration diagram for Allocators:



Classes

- struct __gnu_cxx::__alloc_traits< _Alloc, typename >
- class __gnu_cxx::__mt_alloc< _Tp, _Poolp >
- class std::__new_allocator< _Tp >
- class __gnu_cxx::__pool_alloc< _Tp >
- class __gnu_cxx::_ExtPtr_allocator< _Tp >
- class std::allocator< _Tp >
- class std::allocator< void >
- struct std::allocator_traits< _Alloc >
- struct std::allocator_traits< allocator< _Tp > >
- struct std::allocator_traits< allocator< void > >
- class __gnu_cxx::bitmap_allocator< _Tp >
- class __gnu_cxx::debug_allocator< _Alloc >
- class __gnu_cxx::malloc_allocator< _Tp >
- class __gnu_cxx::new_allocator< _Tp >
- class std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >
- class __gnu_cxx::throw_allocator_base< _Tp, _Cond >
- struct std::uses_allocator< typename, typename >

Typedefs

- template<typename _Tp>
  using std::__allocator_base

Functions

- template<typename _Tp, typename _Alloc, typename... _Args>
  constexpr _Tp **std::make_obj_using_allocator** (const _Alloc &__a, _Args &&... __args)
- template<typename _T1, typename _T2>
  constexpr bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept
- template<typename _OutA1, typename _OutA2, typename... _InA>
  bool **operator==** (const scoped_allocator_adaptor< _OutA1, _InA... > &__a, const scoped_allocator_adaptor< _OutA2, _InA... > &__b) noexcept

- template<typename _Tp, typename _Alloc, typename... _Args>
  constexpr _Tp * **std::uninitialized_construct_using_allocator** (_Tp *__p, const _Alloc &__a, _Args &&... _↩
  _args)
- template<_Std_pair _Tp, typename _Alloc>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, _Up &&, _Vp &&) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, const pair< _Up, _Vp > &&) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, const pair< _Up, _Vp > &) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, pair< _Up, _Vp > &&) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, pair< _Up, _Vp > &) noexcept
- template<typename _Tp, typename _Alloc, typename... _Args>
  requires (! _Std_pair<_Tp>)
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &__a, _Args &&... __args) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Tuple1, typename _Tuple2>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &__a, piecewise_construct_t, _Tuple1 &&__x, _Tuple2 &&__y) noexcept

3.16.6.3.1 Detailed Description

Classes encapsulating memory operations.

3.16.6.3.2 Typedef Documentation

__allocator_base

```
template<typename _Tp>
using std::__allocator_base
```

An alias to the base class for std::allocator.
Used to set the std::allocator base class to std::__new_allocator.

Template Parameters

_Tp	Type of allocated object.

3.16.6.3.3 Function Documentation

operator==()

```
template<typename _T1, typename _T2>
bool operator== (
            const allocator< _T1 > & ,
            const allocator< _T2 > & )  [related]
```

Equality comparison for std::allocator objects

Returns

true, for all std::allocator objects.

3.16.6.4 Pointer Abstractions

Collaboration diagram for Pointer Abstractions:



Classes

- class std::auto_ptr< _Tp >
- struct std::default_delete< _Tp >
- struct std::default_delete< _Tp[ ]>
- class std::enable_shared_from_this< _Tp >
- struct std::hash< shared_ptr< _Tp > >
- struct std::hash< unique_ptr< _Tp, _Dp > >
- struct std::owner_less< _Tp >
- struct std::owner_less< shared_ptr< _Tp > >
- struct std::owner_less< void >
- struct std::owner_less< weak_ptr< _Tp > >
- struct std::pointer_traits< _Ptr >
- struct std::pointer_traits< _Tp * >
- class std::shared_ptr< _Tp >
- class std::unique_ptr< _Tp, _Dp >
- class std::unique_ptr< _Tp[ ], _Dp >
- class std::weak_ptr< _Tp >

Functions

- template<typename _Del, typename _Tp, _Lock_policy _Lp>
  _Del * **std::get_deleter** (const __shared_ptr< _Tp, _Lp > &__p) noexcept
- template<typename _Del, typename _Tp>
  _Del * get_deleter (const shared_ptr< _Tp > &__p) noexcept
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>
  std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)
- template<typename _CharT, typename _Traits, typename _Tp, typename _Dp>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const unique_ptr< _Tp, _Dp > &__p)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)

- template<typename _Tp, typename _Dp>
  constexpr bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer, typename unique_ptr< _Up, _Ep >::pointer > **std::operator<=>** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer > **std::operator<=>** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  bool std::operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Del>
  std::shared_ptr (unique_ptr< _Tp, _Del >) -> shared_ptr< _Tp >
- template<typename _Tp>
  std::shared_ptr (weak_ptr< _Tp >) -> shared_ptr< _Tp >
- template<typename _Tp, typename _Dp>
  enable_if<!__is_swappable< _Dp >::value >::type **std::swap** (unique_ptr< _Tp, _Dp > &, unique_ptr< _Tp, _Dp > &)=delete
- template<typename _Tp, typename _Dp>
  constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept
- template<typename _Tp>
  void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept
- template<typename _Tp>
  constexpr _Tp * std::to_address (_Tp *__ptr) noexcept
- template<typename _Ptr>
  constexpr auto std::to_address (const _Ptr &__ptr) noexcept
- template<typename _Tp>
  std::weak_ptr (shared_ptr< _Tp >) -> weak_ptr< _Tp >

- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_is_lock_free (const __shared_ptr< _Tp, _Lp > *)
- template<typename _Tp>
  bool std::atomic_is_lock_free (const shared_ptr< _Tp > *__p)

- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order)

- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_load (const shared_ptr< _Tp > *__p)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_load_explicit (const __shared_ptr< _Tp, _Lp > *__p, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_load (const __shared_ptr< _Tp, _Lp > *__p)

- template<typename _Tp>
  void std::atomic_store_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)
- template<typename _Tp>
  void std::atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp, _Lock_policy _Lp>
  void std::atomic_store_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  void std::atomic_store (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)

- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_exchange_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)
- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_exchange_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_exchange (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)

- template<typename _Tp>
  bool std::atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order, memory_order)
- template<typename _Tp>
  bool std::atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp>
  bool std::atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  bool std::atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_strong_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)
- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_weak_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)

- template<typename _Tp, typename _Up>
  bool operator== (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept

- template<typename _Tp>
  bool operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp>
  void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > reinterpret_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > static_pointer_cast (shared_ptr< _Up > &&__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > const_pointer_cast (shared_ptr< _Up > &&__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > dynamic_pointer_cast (shared_ptr< _Up > &&__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > reinterpret_pointer_cast (shared_ptr< _Up > &&__r) noexcept

- template<typename _Tp, typename _Alloc, typename... _Args>
  shared_ptr< _NonArray< _Tp > > allocate_shared (const _Alloc &__a, _Args &&... __args)
- template<typename _Tp, typename... _Args>
  shared_ptr< _NonArray< _Tp > > make_shared (_Args &&... __args)

3.16.6.4.1 Detailed Description

Smart pointers, etc.

3.16.6.4.2 Function Documentation

allocate_shared()

```
template<typename _Tp, typename _Alloc, typename...  _Args>
shared_ptr< _NonArray< _Tp > > allocate_shared (
            const _Alloc & __a,
            _Args &&...  __args)  [related]
```

Create an object that is owned by a shared_ptr.

Parameters

__a	An allocator.
__args	Arguments for the *_Tp* object's constructor.

Returns

A shared_ptr that owns the newly created object.

Exceptions

An	exception thrown from *_Alloc::allocate* or from the constructor of *_Tp*.

A copy of *__a* will be used to allocate memory for the shared_ptr and the new object.

atomic_compare_exchange_strong() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_strong (
             __shared_ptr< _Tp, _Lp > * __p,
             __shared_ptr< _Tp, _Lp > * __v,
             __shared_ptr< _Tp, _Lp > __w)  [inline]
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if `*__p` was equivalent to `*__v`, false otherwise.

The memory order for failure shall not be `memory_order_release` or `memory_order_acq_rel`.
References move().

atomic_compare_exchange_strong() [2/2]

```
template<typename _Tp>
bool std::atomic_compare_exchange_strong (
             shared_ptr< _Tp > * __p,
             shared_ptr< _Tp > * __v,
             shared_ptr< _Tp > __w)  [inline]
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if ∗__p was equivalent to ∗__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_compare_exchange_strong_explicit() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_strong_explicit (
             __shared_ptr< _Tp, _Lp > * __p,
             __shared_ptr< _Tp, _Lp > * __v,
             __shared_ptr< _Tp, _Lp > __w,
             memory_order ,
             memory_order )
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if ∗__p was equivalent to ∗__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_compare_exchange_strong_explicit() [2/2]

```
template<typename _Tp>
bool std::atomic_compare_exchange_strong_explicit (
             shared_ptr< _Tp > * __p,
             shared_ptr< _Tp > * __v,
             shared_ptr< _Tp > __w,
             memory_order ,
             memory_order )
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if *__p was equivalent to *__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_compare_exchange_weak() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_weak (
             __shared_ptr< _Tp, _Lp > * __p,
             __shared_ptr< _Tp, _Lp > * __v,
             __shared_ptr< _Tp, _Lp > __w)  [inline]
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if *__p was equivalent to *__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_compare_exchange_weak() [2/2]

```
template<typename _Tp>
bool std::atomic_compare_exchange_weak (
             shared_ptr< _Tp > * __p,
             shared_ptr< _Tp > * __v,
             shared_ptr< _Tp > __w)  [inline]
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if *__p was equivalent to *__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_compare_exchange_weak_explicit() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_compare_exchange_weak_explicit (
             __shared_ptr< _Tp, _Lp > * __p,
             __shared_ptr< _Tp, _Lp > * __v,
             __shared_ptr< _Tp, _Lp > __w,
             memory_order __success,
             memory_order __failure)  [inline]
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if *__p was equivalent to *__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_compare_exchange_weak_explicit() [2/2]

```
template<typename _Tp>
bool std::atomic_compare_exchange_weak_explicit (
             shared_ptr< _Tp > * __p,
             shared_ptr< _Tp > * __v,
             shared_ptr< _Tp > __w,
             memory_order __success,
             memory_order __failure)  [inline]
```

Atomic compare-and-swap for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _v	A non-null pointer to a shared_ptr object.
_↩ _w	A non-null pointer to a shared_ptr object.

Returns

True if *__p was equivalent to *__v, false otherwise.

The memory order for failure shall not be memory_order_release or memory_order_acq_rel.
References move().

atomic_exchange() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_exchange (
             __shared_ptr< _Tp, _Lp > * __p,
             __shared_ptr< _Tp, _Lp > __r)  [inline]
```

Atomic exchange for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _r	New value to store in *__p.

Returns

The original value of *__p

References move().

atomic_exchange() [2/2]

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_exchange (
             shared_ptr< _Tp > * __p,
             shared_ptr< _Tp > __r)  [inline]
```

Atomic exchange for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _r	New value to store in *__p.

Returns

The original value of *__p

References move().

atomic_exchange_explicit() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_exchange_explicit (
             __shared_ptr< _Tp, _Lp > * __p,
             __shared_ptr< _Tp, _Lp > __r,
             memory_order )  [inline]
```

Atomic exchange for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _r	New value to store in *__p.

Returns

The original value of *__p

atomic_exchange_explicit() [2/2]

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_exchange_explicit (
             shared_ptr< _Tp > * __p,
             shared_ptr< _Tp > __r,
             memory_order )  [inline]
```

Atomic exchange for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _r	New value to store in *__p.

Returns

The original value of *__p

atomic_is_lock_free() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
bool std::atomic_is_lock_free (
             const __shared_ptr< _Tp, _Lp > * )  [inline]
```

Report whether shared_ptr atomic operations are lock-free.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.

Returns

True if atomic access to *__p is lock-free, false otherwise.

atomic_is_lock_free() [2/2]

```
template<typename _Tp>
bool std::atomic_is_lock_free (
            const shared_ptr< _Tp > * __p)  [inline]
```

Report whether shared_ptr atomic operations are lock-free.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.

Returns

True if atomic access to *__p is lock-free, false otherwise.

atomic_load() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_load (
            const __shared_ptr< _Tp, _Lp > * __p)  [inline]
```

Atomic load for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.

Returns

*__p

The memory order shall not be `memory_order_release` or `memory_order_acq_rel`.

atomic_load() [2/2]

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_load (
            const shared_ptr< _Tp > * __p)  [inline]
```

Atomic load for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.

Returns

*__p

The memory order shall not be memory_order_release or memory_order_acq_rel.

atomic_load_explicit() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::atomic_load_explicit (
            const __shared_ptr< _Tp, _Lp > * __p,
            memory_order )  [inline]
```

Atomic load for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.

Returns

*__p

The memory order shall not be memory_order_release or memory_order_acq_rel.

atomic_load_explicit() [2/2]

```
template<typename _Tp>
shared_ptr< _Tp > std::atomic_load_explicit (
            const shared_ptr< _Tp > * __p,
            memory_order )  [inline]
```

Atomic load for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.

Returns

*__p

The memory order shall not be memory_order_release or memory_order_acq_rel.

atomic_store() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
void std::atomic_store (
```

```
__shared_ptr< _Tp, _Lp > * __p,
__shared_ptr< _Tp, _Lp > __r)  [inline]
```

Atomic store for shared_ptr objects.

Parameters

_p	A non-null pointer to a shared_ptr object.
_r	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.
References move().

atomic_store() [2/2]

```
template<typename _Tp>
void std::atomic_store (
            shared_ptr< _Tp > * __p,
            shared_ptr< _Tp > __r)  [inline]
```

Atomic store for shared_ptr objects.

Parameters

_p	A non-null pointer to a shared_ptr object.
_r	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.
References move().

atomic_store_explicit() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
void std::atomic_store_explicit (
            __shared_ptr< _Tp, _Lp > * __p,
            __shared_ptr< _Tp, _Lp > __r,
            memory_order )  [inline]
```

Atomic store for shared_ptr objects.

Parameters

_p	A non-null pointer to a shared_ptr object.

_↩ _r	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

atomic_store_explicit() [2/2]

```
template<typename _Tp>
void std::atomic_store_explicit (
             shared_ptr< _Tp > * __p,
             shared_ptr< _Tp > __r,
             memory_order )  [inline]
```

Atomic store for shared_ptr objects.

Parameters

_↩ _p	A non-null pointer to a shared_ptr object.
_↩ _r	The value to store.

The memory order shall not be `memory_order_acquire` or `memory_order_acq_rel`.

const_pointer_cast() [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > const_pointer_cast (
             const shared_ptr< _Up > & __r)  [related]
```

Convert type of `shared_ptr`, via `const_cast`.

const_pointer_cast() [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > const_pointer_cast (
             shared_ptr< _Up > && __r)  [related]
```

Convert type of `shared_ptr` rvalue, via `const_cast`.

Since

C++20

dynamic_pointer_cast() [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > dynamic_pointer_cast (
             const shared_ptr< _Up > & __r)  [related]
```

Convert type of `shared_ptr`, via `dynamic_cast`.

dynamic_pointer_cast() [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > dynamic_pointer_cast (
            shared_ptr< _Up > && __r)  [related]
```

Convert type of `shared_ptr` rvalue, via `dynamic_cast`.

Since

C++20

get_deleter()

```
template<typename _Del, typename _Tp>
_Del * get_deleter (
            const shared_ptr< _Tp > & __p)  [related]
```

20.7.2.2.10 shared_ptr get_deleter
If `__p` has a deleter of type `_Del`, return a pointer to it.

make_shared()

```
template<typename _Tp, typename... _Args>
shared_ptr< _NonArray< _Tp > > make_shared (
            _Args &&... __args)  [related]
```

Create an object that is owned by a shared_ptr.

Parameters

__args	Arguments for the *_Tp* object's constructor.

Returns

A shared_ptr that owns the newly created object.

Exceptions

std::bad_alloc,or	an exception thrown from the constructor of *_Tp*.

operator<() [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator< (
            const unique_ptr< _Tp, _Dp > & __x,
            const unique_ptr< _Up, _Ep > & __y)  [inline], [nodiscard], [constexpr]
```

Relational operator for unique_ptr objects, compares the owned pointers.
References std::unique_ptr< _Tp, _Dp >::get().

operator<() [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator< (
            const unique_ptr< _Tp, _Dp > & __x,
            nullptr_t )  [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr
References std::unique_ptr< _Tp, _Dp >::get().

operator<() [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator< (
            nullptr_t ,
            const unique_ptr< _Tp, _Dp > & __x)  [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr
References std::unique_ptr< _Tp, _Dp >::get().

operator<<() [1/2]

```
template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>
std::basic_ostream< _Ch, _Tr > & operator<< (
            std::basic_ostream< _Ch, _Tr > & __os,
            const __shared_ptr< _Tp, _Lp > & __p)  [related]
```

Write the stored pointer to an ostream.

operator<<() [2/2]

```
template<typename _CharT, typename _Traits, typename _Tp, typename _Dp>
basic_ostream< _CharT, _Traits > & operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const unique_ptr< _Tp, _Dp > & __p)  [related]
```

Stream output operator for unique_ptr.

Since

C++20

operator<=() [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator<= (
            const unique_ptr< _Tp, _Dp > & __x,
            const unique_ptr< _Up, _Ep > & __y)  [inline], [nodiscard], [constexpr]
```

Relational operator for unique_ptr objects, compares the owned pointers.

operator<=() [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator<= (
            const unique_ptr< _Tp, _Dp > & __x,
            nullptr_t )  [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr

operator<=() [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator<= (
            nullptr_t ,
            const unique_ptr< _Tp, _Dp > & __x)  [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr

operator==() [1/4]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator== (
            const unique_ptr< _Tp, _Dp > & __x,
            const unique_ptr< _Up, _Ep > & __y)  [inline], [nodiscard], [constexpr]
```

Equality operator for unique_ptr objects, compares the owned pointers.
References std::unique_ptr< _Tp, _Dp >::get().

operator==() [2/4]

```
template<typename _Tp, typename _Dp>
bool std::operator== (
            const unique_ptr< _Tp, _Dp > & __x,
            nullptr_t )  [inline], [nodiscard], [constexpr], [noexcept]
```

unique_ptr comparison with nullptr

operator==() [3/4]

```
template<typename _Tp, typename _Up>
bool operator== (
            const shared_ptr< _Tp > & __a,
            const shared_ptr< _Up > & __b)  [related]
```

Equality operator for shared_ptr objects, compares the stored pointers.

operator==() [4/4]

```
template<typename _Tp>
bool operator== (
            const shared_ptr< _Tp > & __a,
            nullptr_t )  [related]
```

shared_ptr comparison with nullptr

operator>() [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator> (
            const unique_ptr< _Tp, _Dp > & __x,
            const unique_ptr< _Up, _Ep > & __y)  [inline], [nodiscard], [constexpr]
```

Relational operator for unique_ptr objects, compares the owned pointers.

operator>() [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator> (
            const unique_ptr< _Tp, _Dp > & __x,
            nullptr_t )  [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr
References std::unique_ptr< _Tp, _Dp >::get().

operator>() [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator> (
            nullptr_t ,
            const unique_ptr< _Tp, _Dp > & __x)  [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr
References std::unique_ptr< _Tp, _Dp >::get().

operator>=() [1/3]

```
template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
bool std::operator>= (
            const unique_ptr< _Tp, _Dp > & __x,
            const unique_ptr< _Up, _Ep > & __y)  [inline], [nodiscard], [constexpr]
```

Relational operator for unique_ptr objects, compares the owned pointers.

operator>=() [2/3]

```
template<typename _Tp, typename _Dp>
bool std::operator>= (
            const unique_ptr< _Tp, _Dp > & __x,
            nullptr_t )  [inline], [nodiscard], [constexpr]
```

unique_ptr comparison with nullptr

operator>=() [3/3]

```
template<typename _Tp, typename _Dp>
bool std::operator>= (
            nullptr_t ,
            const unique_ptr< _Tp, _Dp > & __x)  [inline], [nodiscard]
```

unique_ptr comparison with nullptr

reinterpret_pointer_cast() [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > reinterpret_pointer_cast (
            const shared_ptr< _Up > & __r)  [related]
```

Convert type of `shared_ptr`, via `reinterpret_cast`.

Since

C++17

reinterpret_pointer_cast() [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > reinterpret_pointer_cast (
            shared_ptr< _Up > && __r)  [related]
```

Convert type of `shared_ptr` rvalue, via `reinterpret_cast`.

Since

C++20

static_pointer_cast() [1/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > static_pointer_cast (
            const shared_ptr< _Up > & __r)  [related]
```

Convert type of `shared_ptr`, via `static_cast`.

static_pointer_cast() [2/2]

```
template<typename _Tp, typename _Up>
shared_ptr< _Tp > static_pointer_cast (
            shared_ptr< _Up > && __r)  [related]
```

Convert type of `shared_ptr` rvalue, via `static_cast`.

Since

C++20

swap() [1/3]

```
template<typename _Tp>
void swap (
            shared_ptr< _Tp > & __a,
            shared_ptr< _Tp > & __b)  [related]
```

Swap overload for shared_ptr.

swap() [2/3]

```
template<typename _Tp, typename _Dp>
enable_if< __is_swappable< _Dp >::value >::type swap (
            unique_ptr< _Tp, _Dp > & __x,
            unique_ptr< _Tp, _Dp > & __y)  [related]
```

Swap overload for unique_ptr.

swap() [3/3]

```
template<typename _Tp>
void swap (
            weak_ptr< _Tp > & __a,
            weak_ptr< _Tp > & __b)  [related]
```

Swap overload for weak_ptr.

to_address() [1/2]

```
template<typename _Tp>
_Tp * std::to_address (
            _Tp * __ptr)  [constexpr], [noexcept]
```

Obtain address referenced by a pointer to an object.

Parameters

__ptr	A pointer to an object

Returns

`__ptr`

Referenced by std::basic_string< _CharT >::assign(), find(), to_address(), uninitialized_copy(), uninitialized_fill(), and uninitialized_fill_n().

to_address() [2/2]

```
template<typename _Ptr>
auto std::to_address (
            const _Ptr & __ptr)  [constexpr], [noexcept]
```

Obtain address referenced by a pointer to an object.

Parameters

__ptr	A pointer to an object

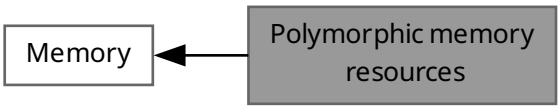
Returns

`pointer_traits<_Ptr>::to_address(__ptr)` if that expression is well-formed, otherwise `to_address(__ptr.operator->())`.

References to_address().

3.16.6.5 Polymorphic memory resources

Collaboration diagram for Polymorphic memory resources:



Files

- file memory_resource

Classes

- struct std::allocator_traits< pmr::polymorphic_allocator< _Tp > >
- class std::pmr::memory_resource
- class std::pmr::monotonic_buffer_resource
- class std::pmr::polymorphic_allocator< _Tp >
- struct std::pmr::pool_options
- class std::pmr::unsynchronized_pool_resource

Functions

- memory_resource * std::pmr::new_delete_resource () noexcept

3.16.6.5.1 Detailed Description

Since

C++17

Memory resources are classes that implement the `std::pmr::memory_resource` interface for allocating and deallocating memory. Unlike traditional C++ allocators, memory resources are not value types and are used via pointers to the abstract base class. They are only responsible for allocating and deallocating, not for construction and destruction of objects. As a result, memory resources just allocate raw memory as type `void*` and are not templates that allocate/deallocate and construct/destroy a specific type.
The class template `std::pmr::polymorphic_allocator` is an allocator that uses a memory resource for its allocations.

3.16.6.5.2 Function Documentation

new_delete_resource()

```
memory_resource * std::pmr::new_delete_resource ()  [nodiscard], [noexcept]
```

A pmr::memory_resource that uses `new` to allocate memory.

Since

C++17

3.16.7 Metaprogramming

Collaboration diagram for Metaprogramming:



Topics

- Variable Templates

Classes

- struct std::tr2::__reflection_typelist< _Elements >
- struct std::tr2::__reflection_typelist< _First, _Rest... >
- struct std::tr2::__reflection_typelist<>
- struct std::add_const< _Tp >
- struct std::add_cv< _Tp >
- struct std::add_lvalue_reference< _Tp >
- struct std::add_pointer< _Tp >
- struct std::add_rvalue_reference< _Tp >
- struct std::add_volatile< _Tp >
- struct std::aligned_storage< _Len, _Align >
- struct std::aligned_union< _Len, _Types >
- struct std::alignment_of< _Tp >
- struct std::tr2::bases< _Tp >
- struct std::common_type< _Tp >

- struct std::conditional< _Cond, _Iftrue, _Iffalse >
- struct std::tr2::direct_bases< _Tp >
- struct std::enable_if< bool, _Tp >
- struct std::extent< typename, _Uint >
- struct std::has_virtual_destructor< _Tp >
- struct std::integral_constant< _Tp, __v >
- struct std::is_abstract< _Tp >
- struct std::is_arithmetic< _Tp >
- struct std::is_array< _Tp >
- struct std::is_assignable< _Tp, _Up >
- struct std::is_base_of< _Base, _Derived >
- struct std::is_class< _Tp >
- struct std::is_compound< _Tp >
- struct std::is_const< _Tp >
- struct std::is_constructible< _Tp, _Args >
- struct std::is_copy_assignable< _Tp >
- struct std::is_copy_constructible< _Tp >
- struct std::is_default_constructible< _Tp >
- struct std::is_destructible< _Tp >
- struct std::is_empty< _Tp >
- struct std::is_enum< _Tp >
- struct std::is_floating_point< _Tp >
- struct std::is_function< _Tp >
- struct std::is_fundamental< _Tp >
- struct std::is_integral< _Tp >
- struct std::is_layout_compatible< _Tp, _Up >
- struct std::is_literal_type< _Tp >
- struct std::is_lvalue_reference< typename >
- struct std::is_member_function_pointer< _Tp >
- struct std::is_member_object_pointer< _Tp >
- struct std::is_member_pointer< _Tp >
- struct std::is_move_assignable< _Tp >
- struct std::is_move_constructible< _Tp >
- struct std::is_nothrow_assignable< _Tp, _Up >
- struct std::is_nothrow_constructible< _Tp, _Args >
- struct std::is_nothrow_copy_assignable< _Tp >
- struct std::is_nothrow_copy_constructible< _Tp >
- struct std::is_nothrow_default_constructible< _Tp >
- struct std::is_nothrow_destructible< _Tp >
- struct std::is_nothrow_move_assignable< _Tp >
- struct std::is_nothrow_move_constructible< _Tp >
- struct std::is_object< _Tp >
- struct std::is_pod< _Tp >
- struct std::is_pointer< _Tp >
- struct std::is_pointer_interconvertible_base_of< _Base, _Derived >
- struct std::is_polymorphic< _Tp >
- struct std::is_reference< _Tp >
- struct std::is_rvalue_reference< typename >
- struct std::is_same< _Tp, _Up >
- struct std::is_scalar< _Tp >
- struct std::is_signed< _Tp >

- struct std::is_standard_layout< _Tp >
- struct std::is_trivial< _Tp >
- struct std::is_trivially_assignable< _Tp, _Up >
- struct std::is_trivially_constructible< _Tp, _Args >
- struct std::is_trivially_copy_assignable< _Tp >
- struct std::is_trivially_copy_constructible< _Tp >
- struct std::is_trivially_copyable< _Tp >
- struct std::is_trivially_default_constructible< _Tp >
- struct std::is_trivially_destructible< _Tp >
- struct std::is_trivially_move_assignable< _Tp >
- struct std::is_trivially_move_constructible< _Tp >
- struct std::is_union< _Tp >
- struct std::is_unsigned< _Tp >
- struct std::is_void< _Tp >
- struct std::is_volatile< _Tp >
- struct std::make_signed< _Tp >
- struct std::make_unsigned< _Tp >
- struct std::rank< _Tp >
- struct std::remove_all_extents< _Tp >
- struct std::remove_const< _Tp >
- struct std::remove_cv< _Tp >
- struct std::remove_extent< _Tp >
- struct std::remove_pointer< _Tp >
- struct std::remove_reference< _Tp >
- struct std::remove_volatile< _Tp >
- struct std::result_of< _Signature >
- struct std::underlying_type< _Tp >

Typedefs

- template<bool _Cond, typename _If, typename _Else>
  using **std::__conditional_t**
- template<bool _Cond, typename _Tp = void>
  using **std::__enable_if_t**
- template<typename _ToElementType, typename _FromElementType>
  using **std::__is_array_convertible**
- using **std::_Build_index_tuple< _Num >::__type**
- template<typename, size_t... _Indices>
  using **std::_Build_index_tuple< _Num >::_IdxTuple**
- template<typename _Tp>
  using std::add_lvalue_reference_t
- template<typename _Tp>
  using std::add_pointer_t
- template<typename _Tp>
  using std::add_rvalue_reference_t
- template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
  using std::aligned_storage_t
- template<size_t _Len, typename... _Types>
  using **std::aligned_union_t**
- template<typename... _Tp>
  using std::common_type_t

- template<bool _Cond, typename _Iftrue, typename _Iffalse>
  using std::conditional_t
- template<typename _Tp>
  using std::decay_t
- template<bool _Cond, typename _Tp = void>
  using std::enable_if_t
- using std::false_type
- template<typename _Tp>
  using std::make_signed_t
- template<typename _Tp>
  using std::make_unsigned_t
- template<typename _Tp>
  using std::remove_all_extents_t
- template<typename _Tp>
  using std::remove_extent_t
- template<typename _Tp>
  using std::remove_pointer_t
- template<typename _Tp>
  using std::remove_reference_t
- template<typename _Tp>
  using std::result_of_t
- using std::true_type
- template<typename _Tp, typename>
  using **std::__conditional< bool >::type**
- template<typename, typename _Up>
  using **std::__conditional< false >::type**
- using **std::__is_implicitly_default_constructible_impl< _Tp >::type**
- using **std::add_const< _Tp >::type**
- using **std::add_cv< _Tp >::type**
- using **std::add_lvalue_reference< _Tp >::type**
- using **std::add_pointer< _Tp >::type**
- using **std::add_rvalue_reference< _Tp >::type**
- using **std::add_volatile< _Tp >::type**
- using std::aligned_union< _Len, _Types >::type
- using **std::common_reference< _Tp0 >::type**
- using **std::conditional< _Cond, _Iftrue, _Iffalse >::type**
- using **std::conditional< false, _Iftrue, _Iffalse >::type**
- using **std::decay< _Tp >::type**
- using **std::enable_if< true, _Tp >::type**
- using **std::integral_constant< _Tp, __v >::type**
- using **std::make_signed< _Tp >::type**
- using **std::make_unsigned< _Tp >::type**
- using **std::remove_all_extents< _Tp >::type**
- using **std::remove_const< _Tp >::type**
- using **std::remove_const< _Tp const >::type**
- using **std::remove_cv< _Tp >::type**
- using **std::remove_extent< _Tp >::type**
- using **std::remove_pointer< _Tp >::type**
- using **std::remove_reference< _Tp >::type**
- using **std::remove_volatile< _Tp >::type**
- using **std::remove_volatile< _Tp volatile >::type**
- template<typename _Tp>
  using std::underlying_type_t
- using **std::integral_constant< _Tp, __v >::value_type**

Functions

- template<typename _Tp>
  static void **std::__do_is_implicitly_default_constructible_impl::__helper** (const _Tp &)
- static false_type **std::__do_is_implicitly_default_constructible_impl::__test** (...)
- template<typename _Tp>
  static true_type **std::__do_is_implicitly_default_constructible_impl::__test** (const _Tp &, decltype(__helper< const _Tp & >({})) *=0)
- constexpr **std::integral_constant< _Tp, __v >::operator value_type** () const noexcept
- template<typename _Tp>
  constexpr _Require< __not_< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable< _Tp > > std::swap (_Tp &, _Tp &) noexcept(__and_< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp > >::value)
- template<typename _Tp, size_t _Nm>
  constexpr __enable_if_t< __is_swappable< _Tp >::value > std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]↩) noexcept(__is_nothrow_swappable< _Tp >::value)

Variables

- unsigned char **std::aligned_storage< _Len, _Align >::type::__data** [_Len]
- template<typename _Tp, typename... _Args>
  constexpr bool **std::__is_nothrow_new_constructible**
- static const size_t **std::__strictest_alignment< _Types >::_S_alignment**
- static const size_t **std::__strictest_alignment< _Tp, _Types... >::_S_alignment**
- static const size_t **std::__strictest_alignment< _Types >::_S_size**
- static const size_t **std::__strictest_alignment< _Tp, _Types... >::_S_size**
- template<size_t _Len, typename... _Types>
  const size_t **std::aligned_union< _Len, _Types... >::alignment_value**
- static const size_t std::aligned_union< _Len, _Types >::alignment_value
- static constexpr _Tp **std::integral_constant< _Tp, __v >::value**

- template<typename... _Tp>
  using std::common_reference_t
- template<typename _Tp, typename _Up>
  constexpr bool std::is_layout_compatible_v
- template<typename _Base, typename _Derived>
  constexpr bool std::is_pointer_interconvertible_base_of_v
- template<typename _S1, typename _S2, typename _M1, typename _M2>
  constexpr bool std::is_corresponding_member (_M1 _S1::*__m1, _M2 _S2::*__m2) noexcept
- template<typename _Tp, typename _Mem>
  constexpr bool std::is_pointer_interconvertible_with_class (_Mem _Tp::*__mp) noexcept

3.16.7.1 Detailed Description

Template utilities for compile-time introspection and modification, including type classification traits, type property inspection traits and type transformation traits.

Since

C++11

3.16.7.2 Typedef Documentation

add_lvalue_reference_t

```
template<typename _Tp>
using std::add_lvalue_reference_t
```

Alias template for add_lvalue_reference.

add_pointer_t

```
template<typename _Tp>
using std::add_pointer_t
```

Alias template for add_pointer.

add_rvalue_reference_t

```
template<typename _Tp>
using std::add_rvalue_reference_t
```

Alias template for add_rvalue_reference.

aligned_storage_t

```
template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
using std::aligned_storage_t
```

Alias template for aligned_storage.

common_reference_t

```
template<typename... _Tp>
using std::common_reference_t
```

Since

C++20

common_type_t

```
template<typename... _Tp>
using std::common_type_t
```

Alias template for common_type.

conditional_t

```
template<bool _Cond, typename _Iftrue, typename _Iffalse>
using std::conditional_t
```

Alias template for conditional.

decay_t

```
template<typename _Tp>
using std::decay_t
```

Alias template for decay.

enable_if_t

```
template<bool _Cond, typename _Tp = void>
using std::enable_if_t
```

Alias template for enable_if.

false_type

```
using std::false_type
```
The type used as a compile-time boolean with false value.

make_signed_t

```
template<typename _Tp>
using std::make_signed_t
```
Alias template for make_signed.

make_unsigned_t

```
template<typename _Tp>
using std::make_unsigned_t
```
Alias template for make_unsigned.

remove_all_extents_t

```
template<typename _Tp>
using std::remove_all_extents_t
```
Alias template for remove_all_extents.

remove_extent_t

```
template<typename _Tp>
using std::remove_extent_t
```
Alias template for remove_extent.

remove_pointer_t

```
template<typename _Tp>
using std::remove_pointer_t
```
Alias template for remove_pointer.

remove_reference_t

```
template<typename _Tp>
using std::remove_reference_t
```
Alias template for remove_reference.

result_of_t

```
template<typename _Tp>
using std::result_of_t
```
Alias template for result_of.

true_type

```
using std::true_type
```
The type used as a compile-time boolean with true value.

type

```
template<size_t _Len, typename...  _Types>
using std::aligned_union< _Len, _Types >::type
```
The storage.

underlying_type_t

```
template<typename _Tp>
using std::underlying_type_t
```
Alias template for underlying_type.

3.16.7.3 Function Documentation

is_corresponding_member()

```
template<typename _S1, typename _S2, typename _M1, typename _M2>
bool std::is_corresponding_member (
             _M1 _S1::* __m1,
             _M2 _S2::* __m2)  [constexpr], [noexcept]
```

Since

C++20

is_pointer_interconvertible_with_class()

```
template<typename _Tp, typename _Mem>
bool std::is_pointer_interconvertible_with_class (
             _Mem _Tp::* __mp)  [constexpr], [noexcept]
```
True if `__mp` points to the first member of a standard-layout type.

Returns

true if `s.*__mp` is pointer-interconvertible with `s`

Since

C++20

swap() [1/2]

```
template<typename _Tp>
_Require< __not_< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable<
_Tp > > std::swap (
             _Tp & __a,
             _Tp & __b)  [inline], [constexpr], [noexcept]
```
Swaps two values.

Parameters

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

swap() [2/2]

```
template<typename _Tp, size_t _Nm>
__enable_if_t< __is_swappable< _Tp >::value > std::swap (
            _Tp(&) __a[_Nm],
            _Tp(&) __b[_Nm])  [inline], [constexpr], [noexcept]
```

Swap the contents of two arrays.

3.16.7.4 Variable Documentation

alignment_value

```
template<size_t _Len, typename...  _Types>
const size_t std::aligned_union< _Len, _Types >::alignment_value  [static]
```

The value of the strictest alignment of _Types.

is_layout_compatible_v

```
template<typename _Tp, typename _Up>
bool std::is_layout_compatible_v  [constexpr]
```

Since

C++20

is_pointer_interconvertible_base_of_v

```
template<typename _Base, typename _Derived>
bool std::is_pointer_interconvertible_base_of_v  [constexpr]
```

Since

C++20

3.16.7.5 Variable Templates

Collaboration diagram for Variable Templates:



Variables

- template<typename _Tp, typename _Up>
  constexpr strong_ordering std::type_order_v
- template<typename _Tp, typename _Up>
  constexpr bool std::is_layout_compatible_v
- template<typename _Base, typename _Derived>
  constexpr bool std::is_pointer_interconvertible_base_of_v

3.16.7.5.1 Detailed Description

Since

C++17

Each trait in `<type_traits>` that has a `value` static data member is also available as a variable template, using the same name with a `_v` suffix. For example, `std::is_integral_v<T>` is a `bool` constant with the same value as `std::is_integral<T>::value`.
In some cases, using the variable template instead of the `value` member avoids instantiating the class template for the trait. This gives the compiler less work to do and can make your code compile a little faster.

3.16.7.5.2 Variable Documentation

is_layout_compatible_v

```
template<typename _Tp, typename _Up>
bool std::is_layout_compatible_v  [constexpr]
```

Since

C++20

is_pointer_interconvertible_base_of_v

```
template<typename _Base, typename _Derived>
bool std::is_pointer_interconvertible_base_of_v  [constexpr]
```

Since

C++20

type_order_v

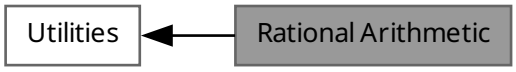
```
template<typename _Tp, typename _Up>
strong_ordering std::type_order_v  [inline], [constexpr]
```

Since

C++26

3.16.8 Rational Arithmetic

Collaboration diagram for Rational Arithmetic:



Files

- file ratio

Classes

- struct std::ratio< _Num, _Den >
- struct std::ratio_equal< _R1, _R2 >
- struct std::ratio_greater< _R1, _R2 >
- struct std::ratio_greater_equal< _R1, _R2 >
- struct std::ratio_less< _R1, _R2 >
- struct std::ratio_less_equal< _R1, _R2 >
- struct std::ratio_not_equal< _R1, _R2 >

Typedefs

- using **std::atto**
- using **std::centi**
- using **std::deca**
- using **std::deci**
- using **std::exa**
- using **std::femto**
- using **std::giga**
- using **std::hecto**
- using **std::kilo**
- using **std::mega**
- using **std::micro**
- using **std::milli**
- using **std::nano**
- using **std::peta**
- using **std::pico**
- template<typename _R1, typename _R2>
  using std::ratio_add
- template<typename _R1, typename _R2>
  using std::ratio_divide
- template<typename _R1, typename _R2>
  using std::ratio_multiply
- template<typename _R1, typename _R2>
  using std::ratio_subtract
- using **std::tera**

Variables

- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_greater_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_greater_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_less_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_less_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_not_equal_v**

3.16.8.1 Detailed Description

Compile time representation of finite rational numbers.

3.16.8.2 Typedef Documentation

ratio_add

```
template<typename _R1, typename _R2>
using std::ratio_add
```

ratio_add

ratio_divide

```
template<typename _R1, typename _R2>
using std::ratio_divide
```

ratio_divide

ratio_multiply

```
template<typename _R1, typename _R2>
using std::ratio_multiply
```

ratio_multiply

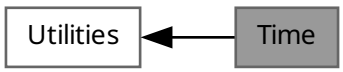
ratio_subtract

```
template<typename _R1, typename _R2>
using std::ratio_subtract
```

ratio_subtract

3.16.9 Time

Collaboration diagram for Time:



Files

- file chrono

Namespaces

- namespace std::chrono
- namespace std::literals::chrono_literals

Classes

- struct std::common_type< chrono::duration< _Rep, _Period > >
- struct std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >
- struct std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >
- struct std::common_type< chrono::time_point< _Clock, _Duration > >
- struct std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >
- struct std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >
- class std::chrono::duration< _Rep, _Period >
- struct std::chrono::duration_values< _Rep >
- class std::chrono::gps_clock
- class std::chrono::hh_mm_ss< _Duration >
- struct std::chrono::steady_clock
- struct std::chrono::system_clock
- class std::chrono::tai_clock
- class std::chrono::time_point< _Clock, _Dur >
- struct std::chrono::treat_as_floating_point< _Rep >
- class std::chrono::tzdb_list
- class std::chrono::utc_clock

Typedefs

- using std::chrono::days
- using **std::chrono::file_clock**
- template<typename _Duration>
  using **std::chrono::file_time**
- using **std::chrono::gps_seconds**
- template<typename _Duration>
  using **std::chrono::gps_time**
- using std::chrono::high_resolution_clock
- using std::chrono::hours
- using **std::chrono::local_days**
- using **std::chrono::local_seconds**
- template<typename _Duration>
  using **std::chrono::local_time**
- using std::chrono::microseconds
- using std::chrono::milliseconds
- using std::chrono::minutes
- using std::chrono::months
- using std::chrono::nanoseconds
- using std::chrono::seconds
- using **std::chrono::sys_days**
- using **std::chrono::sys_seconds**
- template<typename _Duration>
  using **std::chrono::sys_time**
- using **std::chrono::tai_seconds**
- template<typename _Duration>
  using **std::chrono::tai_time**
- using **std::chrono::utc_seconds**
- template<typename _Duration>
  using **std::chrono::utc_time**
- using std::chrono::weeks
- using std::chrono::years
- using **std::chrono::zoned_seconds**

Enumerations

- enum class **choose** { **earliest** , **latest** }

Functions

- constexpr **std::chrono::year_month_day::year_month_day** (const year_month_day_last &__ymdl) noexcept
- template<typename _Rep, typename _Period>
  constexpr enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > > std::chrono::abs (duration< _Rep, _Period > __d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::ceil (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::ceil (const time_point< _Clock, _Dur > &__tp)
- template<typename _DestClock, typename _SourceClock, typename _Duration>
  requires __detail::__clock_convs<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys<_DestClock, _SourceClock, ↩ _Duration> || __detail::__clock_convs_utc<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys_utc<_DestClock, _↩ SourceClock, _Duration> || __detail::__clock_convs_utc_sys<_DestClock, _SourceClock, _Duration>
  auto std::chrono::clock_cast (const time_point< _SourceClock, _Duration > &__t)
- const time_zone * **std::chrono::current_zone** ()
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::duration_cast (const duration< _Rep, _Period > &↩ __d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::floor (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::floor (const time_point< _Clock, _Dur > &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, day &__d, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Rep, typename _Period, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, duration< _Rep, _Period > &__d, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, file_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, gps_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, local_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, month &__m, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *_↩ _offset=nullptr)

- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, month_day &__md, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, sys_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, tai_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, utc_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, weekday &__wd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, year &__y, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, year_month &__ym, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, year_month_day &__ymd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _Duration>
  static utc_time< common_type_t< _Duration, seconds > > **std::chrono::utc_clock::from_sys** (const sys_time< _Duration > &__t)
- template<typename _Duration>
  leap_second_info **std::chrono::get_leap_second_info** (const utc_time< _Duration > &__ut)
- const tzdb & **std::chrono::get_tzdb** ()
- tzdb_list & **std::chrono::get_tzdb_list** ()
- constexpr bool std::chrono::is_am (const hours &__h) noexcept
- constexpr bool std::chrono::is_pm (const hours &__h) noexcept
- template<typename _Duration>
  __detail::__local_time_fmt< _Duration > std::chrono::local_time_format (local_time< _Duration > __time, const string *__abbrev=nullptr, const seconds *__offset_sec=nullptr)
- const time_zone * **std::chrono::locate_zone** (string_view __tz_name)
- constexpr hours std::chrono::make12 (const hours &__h) noexcept
- constexpr hours std::chrono::make24 (const hours &__h, bool __is_pm) noexcept
- constexpr bool **std::chrono::year_month_day::ok** () const noexcept
- constexpr chrono::day std::literals::chrono_literals::operator""d (unsigned long long __d) noexcept
- template<char... _Digits>
  constexpr chrono::hours std::literals::chrono_literals::operator""h ()
- constexpr chrono::duration< long double, ratio< 3600, 1 > > std::literals::chrono_literals::operator""h (long double __hours)

- template<char... _Digits>
  constexpr chrono::minutes std::literals::chrono_literals::operator""min ()
- constexpr chrono::duration< long double, ratio< 60, 1 > > std::literals::chrono_literals::operator""min (long double __mins)
- template<char... _Digits>
  constexpr chrono::milliseconds std::literals::chrono_literals::operator""ms ()
- constexpr chrono::duration< long double, milli > std::literals::chrono_literals::operator""ms (long double __msecs)
- template<char... _Digits>
  constexpr chrono::nanoseconds std::literals::chrono_literals::operator""ns ()
- constexpr chrono::duration< long double, nano > std::literals::chrono_literals::operator""ns (long double __nsecs)
- template<char... _Digits>
  constexpr chrono::seconds std::literals::chrono_literals::operator""s ()
- constexpr chrono::duration< long double > std::literals::chrono_literals::operator""s (long double __secs)
- template<char... _Digits>
  constexpr chrono::microseconds std::literals::chrono_literals::operator""us ()
- constexpr chrono::duration< long double, micro > std::literals::chrono_literals::operator""us (long double __usecs)
- constexpr chrono::year std::literals::chrono_literals::operator""y (unsigned long long __y) noexcept
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const day &__d)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const file_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const gps_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const hh_mm_ss< _Duration > &__hms)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const local_info &__li)
- template<typename _CharT, typename _Traits, typename _Duration>
  requires requires(const sys_time<_Duration>& __st) { __os << __st; }
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const local_time< _Duration > &__lt)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month &__m)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_day &__md)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_day_last &__mdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_weekday &__mwd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_weekday_last &__mwdl)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const sys_days &__dp)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const sys_info &__i)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const sys_time< _Duration > &__tp)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const tai_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const utc_time< _Duration > &__t)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const weekday &__wd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const weekday_indexed &__wdi)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const weekday_last &__wdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year &__y)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month &__ym)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_day &__ymd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_day_last &__ymdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_weekday &__ymwd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_weekday_last &__ymwdl)
- template<typename _CharT, typename _Traits, typename _Duration, typename _TimeZonePtr>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const zoned_time< _Duration, _TimeZonePtr > &__t)
- template<typename _CharT, typename _Traits, typename _Rep, typename _Period>
  basic_ostream< _CharT, _Traits > & std::chrono::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const duration< _Rep, _Period > &__d)
- template<typename _Dur1, typename _TZPtr1, typename _Dur2, typename _TZPtr2>
  bool **std::chrono::operator==** (const zoned_time< _Dur1, _TZPtr1 > &__x, const zoned_time< _Dur2, _TZPtr2 > &__y)
- constexpr weekday_last **std::chrono::weekday::operator[ ]** (last_spec) const noexcept
- constexpr weekday_indexed **std::chrono::weekday::operator[ ]** (unsigned __index) const noexcept

- template<typename _CharT, __detail::__parsable< _CharT > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &_abbrev)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &_abbrev, minutes &__offset)
- template<typename _CharT, typename _Traits = char_traits<_CharT>, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp, minutes &__offset)
- template<typename _CharT, typename _Traits, typename _Alloc, __detail::__parsable< _CharT, _Traits > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev, minutes &__offset)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, minutes &_offset)
- const tzdb & **std::chrono::reload_tzdb** ()
- string **std::chrono::remote_version** ()
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr enable_if_t< __and_< __is_duration< _ToDur >, __not_< treat_as_floating_point< typename _ToDur::rep > > >::value, _ToDur > std::chrono::round (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur > &&!treat_as_floating_point_v< typename _ToDur::rep >, time_point< _Clock, _ToDur > > std::chrono::round (const time_point< _Clock, _Dur > &__tp)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr __enable_if_t< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > > std::chrono::time_point_cast (const time_point< _Clock, _Dur > &__t)
- **std::chrono::zoned_time** () -> zoned_time< seconds >
- template<typename _TimeZonePtrOrName>
  std::chrono::zoned_time (_TimeZonePtrOrName &&) -> zoned_time< seconds, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _TimeZonePtrOrName, typename _Duration>
  std::chrono::zoned_time (_TimeZonePtrOrName &&, local_time< _Duration >, choose=choose::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _TimeZonePtrOrName, typename _Duration>
  std::chrono::zoned_time (_TimeZonePtrOrName &&, sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _Duration, typename _TimeZonePtrOrName, typename _TimeZonePtr2>
  std::chrono::zoned_time (_TimeZonePtrOrName &&, zoned_time< _Duration, _TimeZonePtr2 >, choose=choose::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >

- template<typename _Duration>
  std::chrono::zoned_time (sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds > >

Variables

- constexpr month **std::chrono::April**
- constexpr month **std::chrono::August**
- constexpr month **std::chrono::December**
- constexpr month **std::chrono::February**
- constexpr weekday **std::chrono::Friday**
- template<typename _Tp>
  constexpr bool **std::chrono::is_clock_v**
- template<typename _Tp>
  constexpr bool **std::chrono::is_clock_v**< **_Tp** >
- template<> constexpr bool **std::chrono::is_clock_v**< **file_clock** >
- template<> constexpr bool **std::chrono::is_clock_v**< **gps_clock** >
- template<> constexpr bool **std::chrono::is_clock_v**< **steady_clock** >
- template<> constexpr bool **std::chrono::is_clock_v**< **system_clock** >
- template<> constexpr bool **std::chrono::is_clock_v**< **tai_clock** >
- template<> constexpr bool **std::chrono::is_clock_v**< **utc_clock** >
- constexpr month **std::chrono::January**
- constexpr month **std::chrono::July**
- constexpr month **std::chrono::June**
- constexpr last_spec **std::chrono::last**
- constexpr month **std::chrono::March**
- constexpr month **std::chrono::May**
- constexpr weekday **std::chrono::Monday**
- constexpr month **std::chrono::November**
- constexpr month **std::chrono::October**
- constexpr weekday **std::chrono::Saturday**
- constexpr month **std::chrono::September**
- constexpr weekday **std::chrono::Sunday**
- constexpr weekday **std::chrono::Thursday**
- template<typename _Rep>
  constexpr bool **std::chrono::treat_as_floating_point_v**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v**< **double** >
- template<> constexpr bool **std::chrono::treat_as_floating_point_v**< **float** >
- template<> constexpr bool **std::chrono::treat_as_floating_point_v**< **int** >
- template<> constexpr bool **std::chrono::treat_as_floating_point_v**< **long** >
- template<> constexpr bool **std::chrono::treat_as_floating_point_v**< **long double** >
- template<> constexpr bool **std::chrono::treat_as_floating_point_v**< **long long** >
- constexpr weekday **std::chrono::Tuesday**
- constexpr weekday **std::chrono::Wednesday**

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator-(const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+(const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Rep2, typename _Period>
  constexpr duration< __common_rep_t< _Rep2, _Rep1 >, _Period > std::chrono::operator* (const _Rep1 &__s, const duration< _Rep2, _Period > &__d)
- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::operator/ (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::operator% (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator% (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr auto std::chrono::operator<=> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>
  constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type > std::chrono::operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
  constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > std::chrono::operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr common_type< _Dur1, _Dur2 >::type std::chrono::operator- (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
  constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator+ (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Clock, typename _Dur1, three_way_comparable_with< _Dur1 > _Dur2>
  constexpr auto **std::chrono::operator<=>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _↩
  Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator<** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,
  _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator<=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _↩
  Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock,
  _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator>=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _↩
  Clock, _Dur2 > &__rhs)

3.16.9.1 Detailed Description

Classes and functions for time.

Since

C++11

3.16.9.2 Typedef Documentation

days

```
using std::chrono::days
```

days

high_resolution_clock

```
using std::chrono::high_resolution_clock
```

Highest-resolution clock.
This is the clock "with the shortest tick period." Alias to std::system_clock until higher-than-nanosecond definitions become feasible.

hours

```
using std::chrono::hours
```

hours

microseconds

```
using std::chrono::microseconds
```

microseconds

milliseconds

```
using std::chrono::milliseconds
```

milliseconds

minutes

```
using std::chrono::minutes
```

minutes

months

```
using std::chrono::months
```
months

nanoseconds

```
using std::chrono::nanoseconds
```
nanoseconds

seconds

```
using std::chrono::seconds
```
seconds

weeks

```
using std::chrono::weeks
```
weeks

years

```
using std::chrono::years
```
years

3.16.9.3 Function Documentation

abs()

```
template<typename _Rep, typename _Period>
enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > > std::chrono::abs (
            duration< _Rep, _Period > __d)  [nodiscard], [constexpr]
```
The absolute (non-negative) value of a duration.

Parameters

_↩ _d	A duration with a signed `rep` type.

Returns

A duration of the same type as the argument, with value |d|.

Since

C++17

ceil() [1/2]

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::ceil (
            const duration< _Rep, _Period > & __d)  [nodiscard], [constexpr]
```
Convert a `duration` to type `ToDur` and round up.
If the duration cannot be represented exactly in the result type, returns the closest value that is greater than the argument.

Template Parameters

_ToDur	The result type must be a `duration`.

Parameters

__d	A duration.

Returns

The value of `__d` converted to type `_ToDur`.

Since

C++17

References duration_cast().
Referenced by ceil().

ceil() [2/2]

```
template<typename _ToDur, typename _Clock, typename _Dur>
enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::ceil (
            const time_point< _Clock, _Dur > & __tp)  [nodiscard], [constexpr]
```

Convert a `time_point` to type `ToDur` and round up.
The result is the same time point as measured by the same clock, but using the specified `duration` to represent the time. If the time point cannot be represented exactly in the result type, returns the closest value that is greater than the argument.

Template Parameters

_ToDur	The `duration` type to use for the result.

Parameters

__tp	A time point.

Returns

The value of `__tp` rounded up to the precision of `_ToDur`.

Since

C++17

References ceil().

clock_cast()

```
template<typename _DestClock, typename _SourceClock, typename _Duration>
requires __detail::__clock_convs<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs↩
_sys<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_utc<_DestClock, _SourceClock,
_Duration> || __detail::__clock_convs_sys_utc<_DestClock, _SourceClock, _Duration> || __detail::_↩
_clock_convs_utc_sys<_DestClock, _SourceClock, _Duration>
auto std::chrono::clock_cast (
            const time_point< _SourceClock, _Duration > & __t)  [inline], [nodiscard]
```

Convert a time point to a different clock.

duration_cast()

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::duration_cast (
            const duration< _Rep, _Period > & __d)  [nodiscard], [constexpr]
```

Convert a `duration` to type `ToDur`.
If the duration cannot be represented accurately in the result type, returns the result of integer truncation (i.e., rounded towards zero).

Template Parameters

_ToDur	The result type must be a `duration`.

Parameters

_↩ _d	A duration.

Returns

The value of `__d` converted to type `_ToDur`.

Since

C++11

Referenced by ceil(), floor(), and time_point_cast().

floor() [1/2]

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::floor (
            const duration< _Rep, _Period > & __d)  [nodiscard], [constexpr]
```

Convert a `duration` to type `ToDur` and round down.
If the duration cannot be represented exactly in the result type, returns the closest value that is less than the argument.

Template Parameters

_ToDur	The result type must be a `duration`.

Parameters

_↩ _*d*	A duration.

Returns

The value of __d converted to type _ToDur.

Since

C++17

References duration_cast().
Referenced by floor(), and round().

floor() [2/2]

```
template<typename _ToDur, typename _Clock, typename _Dur>
enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::floor (
            const time_point< _Clock, _Dur > & __tp)  [nodiscard], [constexpr]
```

Convert a time_point to type ToDur and round down.
The result is the same time point as measured by the same clock, but using the specified duration to represent the time. If the time point cannot be represented exactly in the result type, returns the closest value that is less than the argument.

Template Parameters

_ToDur	The duration type to use for the result.

Parameters

_↩ _*tp*	A time point.

Returns

The value of __tp rounded down to the precision of _ToDur.

Since

C++17

References floor().

is_am()

```
bool std::chrono::is_am (
            const hours & __h)  [constexpr], [noexcept]
```

True if a chrono::hours value represents a time before midday.
References is_am().
Referenced by is_am().

is_pm()

```
bool std::chrono::is_pm (
            const hours & __h)  [constexpr], [noexcept]
```

True if a `chrono::hours` value represents a time after midday.
References is_pm().
Referenced by is_pm().

local_time_format()

```
template<typename _Duration>
__detail::__local_time_fmt< _Duration > std::chrono::local_time_format (
            local_time< _Duration > __time,
            const string * __abbrev = nullptr,
            const seconds * __offset_sec = nullptr)  [inline]
```

Return an object that associates timezone info with a local time.
A `chrono::local_time` object has no timezone associated with it. This function creates an object that allows formatting a `local_time` as though it refers to a timezone with the given abbreviated name and offset from UTC.

Since

C++20

make12()

```
hours std::chrono::make12 (
            const hours & __h)  [constexpr], [noexcept]
```

Convert a 24-hour time into a 12-hour time.
References make12().
Referenced by make12().

make24()

```
hours std::chrono::make24 (
            const hours & __h,
            bool __is_pm)  [constexpr], [noexcept]
```

Convert a 12-hour time and p.m. flag into a 24-hour time.
References make24().
Referenced by make24().

operator"""d()

```
chrono::day std::literals::chrono_literals::operator""d (
            unsigned long long __d)  [constexpr], [noexcept]
```

Literal suffix for creating chrono::day objects.

Since

C++20

operator"""h() [1/2]

```
template<char...  _Digits>
chrono::hours std::literals::chrono_literals::operator""h ()  [constexpr]
```

Literal suffix for durations of type `std::chrono::hours`.

operator"""h() [2/2]

```
chrono::duration< long double, ratio< 3600, 1 > > std::literals::chrono_literals::operator""h (
            long double __hours)  [constexpr]
```

Literal suffix for durations representing non-integer hours.

operator"""min() [1/2]

```
template<char... _Digits>
chrono::minutes std::literals::chrono_literals::operator""min ()  [constexpr]
```

Literal suffix for durations of type `std::chrono::minutes`.

operator"""min() [2/2]

```
chrono::duration< long double, ratio< 60, 1 > > std::literals::chrono_literals::operator""min (
            long double __mins)  [constexpr]
```

Literal suffix for durations representing non-integer minutes.

operator"""ms() [1/2]

```
template<char... _Digits>
chrono::milliseconds std::literals::chrono_literals::operator""ms ()  [constexpr]
```

Literal suffix for durations of type `std::chrono::milliseconds`.

operator"""ms() [2/2]

```
chrono::duration< long double, milli > std::literals::chrono_literals::operator""ms (
            long double __msecs)  [constexpr]
```

Literal suffix for durations representing non-integer milliseconds.

operator"""ns() [1/2]

```
template<char... _Digits>
chrono::nanoseconds std::literals::chrono_literals::operator""ns ()  [constexpr]
```

Literal suffix for durations of type `std::chrono::nanoseconds`.

operator"""ns() [2/2]

```
chrono::duration< long double, nano > std::literals::chrono_literals::operator""ns (
            long double __nsecs)  [constexpr]
```

Literal suffix for durations representing non-integer nanoseconds.

operator"""s() [1/2]

```
template<char... _Digits>
chrono::seconds std::literals::chrono_literals::operator""s ()  [constexpr]
```

Literal suffix for durations of type `std::chrono::seconds`.

operator"""s() [2/2]

```
chrono::duration< long double > std::literals::chrono_literals::operator""s (
            long double __secs)  [constexpr]
```

Literal suffix for durations representing non-integer seconds.

operator"""us() [1/2]

```
template<char... _Digits>
chrono::microseconds std::literals::chrono_literals::operator""us () [constexpr]
```

Literal suffix for durations of type `std::chrono::microseconds`.

operator"""us() [2/2]

```
chrono::duration< long double, micro > std::literals::chrono_literals::operator""us (
            long double __usecs) [constexpr]
```

Literal suffix for durations representing non-integer microseconds.

operator"""y()

```
chrono::year std::literals::chrono_literals::operator""y (
            unsigned long long __y) [constexpr], [noexcept]
```

Literal suffix for creating chrono::year objects.

Since

C++20

operator%() [1/2]

```
template<typename _Rep1, typename _Period, typename _Rep2>
duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::↩
operator% (
            const duration< _Rep1, _Period > & __d,
            const _Rep2 & __s) [constexpr]
```

Arithmetic operators for chrono::duration

operator%() [2/2]

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::↩
operator% (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs) [constexpr]
```

Arithmetic operators for chrono::duration

operator*() [1/2]

```
template<typename _Rep1, typename _Period, typename _Rep2>
duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (
            const duration< _Rep1, _Period > & __d,
            const _Rep2 & __s) [related]
```

Arithmetic operators for chrono::duration

operator*() [2/2]

```
template<typename _Rep1, typename _Rep2, typename _Period>
duration< __common_rep_t< _Rep2, _Rep1 >, _Period > std::chrono::operator* (
            const _Rep1 & __s,
            const duration< _Rep2, _Period > & __d) [constexpr]
```

Arithmetic operators for chrono::duration

operator+() [1/3]

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+ (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [related]
```

The sum of two durations.

operator+() [2/3]

```
template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>
time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type > std::↩
chrono::operator+ (
            const duration< _Rep1, _Period1 > & __lhs,
            const time_point< _Clock, _Dur2 > & __rhs)  [constexpr]
```

Adjust a time point forwards by the given duration.

operator+() [3/3]

```
template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator+
(
            const time_point< _Clock, _Dur1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [related]
```

Adjust a time point forwards by the given duration.

operator-() [1/3]

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::↩
operator- (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

The difference between two durations.

operator-() [2/3]

```
template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > std::↩
chrono::operator- (
            const time_point< _Clock, _Dur1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Adjust a time point backwards by the given duration.

operator-() [3/3]

```
template<typename _Clock, typename _Dur1, typename _Dur2>
common_type< _Dur1, _Dur2 >::type std::chrono::operator- (
            const time_point< _Clock, _Dur1 > & __lhs,
            const time_point< _Clock, _Dur2 > & __rhs)  [constexpr]
```

The difference between two time points (as a duration).

operator/() [1/2]

```
template<typename _Rep1, typename _Period, typename _Rep2>
duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::↩
```

```
operator/ (
            const duration< _Rep1, _Period > & __d,
            const _Rep2 & __s)  [constexpr]
```

Arithmetic operators for chrono::duration

operator/() [2/2]

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Arithmetic operators for chrono::duration

operator<()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator< (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Comparisons for chrono::duration

operator<<() [1/3]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::chrono::operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const local_info & __li)
```

Writes a local_info object to an ostream in an unspecified format.

operator<<() [2/3]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::chrono::operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const sys_info & __i)
```

Writes a sys_info object to an ostream in an unspecified format.

operator<<() [3/3]

```
template<typename _CharT, typename _Traits, typename _Rep, typename _Period>
basic_ostream< _CharT, _Traits > & std::chrono::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const duration< _Rep, _Period > & __d)  [inline]
```

Write a `chrono::duration` to an ostream.

Since

C++20

References std::ios_base::flags(), std::ios_base::getloc(), std::basic_ios< _CharT, _Traits >::imbue(), std::move(), operator<<(), and std::ios_base::precision().
Referenced by operator<<().

operator<=()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator<= (
```

```
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Comparisons for chrono::duration

operator<=>()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
auto std::chrono::operator<=> (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Comparisons for chrono::duration

operator==()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool operator== (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [related]
```

Comparisons for chrono::duration

operator>()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator> (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Comparisons for chrono::duration

operator>=()

```
template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
bool std::chrono::operator>= (
            const duration< _Rep1, _Period1 > & __lhs,
            const duration< _Rep2, _Period2 > & __rhs)  [constexpr]
```

Comparisons for chrono::duration

round() [1/2]

```
template<typename _ToDur, typename _Rep, typename _Period>
enable_if_t< __and_< __is_duration< _ToDur >, __not_< treat_as_floating_point< typename _To↩
Dur::rep > > >::value, _ToDur > std::chrono::round (
            const duration< _Rep, _Period > & __d)  [nodiscard], [constexpr]
```

Convert a `duration` to type `ToDur` and round to the closest value.
If the duration cannot be represented exactly in the result type, returns the closest value, rounding ties to even.

Template Parameters

_ToDur	The result type must be a `duration` with a non-floating-point `rep` type.

Parameters

_↩ _d	A duration.

Returns

The value of __d converted to type _ToDur.

Since

C++17

References floor().
Referenced by round().

round() [2/2]

```
template<typename _ToDur, typename _Clock, typename _Dur>
enable_if_t< __is_duration_v< _ToDur > &&!treat_as_floating_point_v< typename _ToDur::rep >,
time_point< _Clock, _ToDur > > std::chrono::round (
            const time_point< _Clock, _Dur > & __tp)  [nodiscard], [constexpr]
```

Convert a time_point to type ToDur and round to the closest value.
The result is the same time point as measured by the same clock, but using the specified duration to represent the time. If the time point cannot be represented exactly in the result type, returns the closest value, rounding ties to even.

Template Parameters

_ToDur	The duration type to use for the result, which must have a non-floating-point rep type.

Parameters

__tp	A time point.

Returns

The value of __tp rounded to the precision _ToDur.

Since

C++17

References round().

time_point_cast()

```
template<typename _ToDur, typename _Clock, typename _Dur>
__enable_if_t< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > > std::chrono::↩
time_point_cast (
            const time_point< _Clock, _Dur > & __t)  [nodiscard], [constexpr]
```

Convert a time_point to use duration type ToDur.
The result is the same time point as measured by the same clock, but using the specified duration to represent the time. If the time point cannot be represented accurately in the result type, returns the result of integer truncation (i.e., rounded towards zero).

Template Parameters

_ToDur	The `duration` type to use for the result.

Parameters

↩ _↩ ↩ _↩ *t*	A time point.

Returns

The value of `__t` converted to use type `_ToDur`.

Since

C++11

References duration_cast().

4 Namespace Documentation

4.1 __gnu_cxx Namespace Reference

Namespaces

- namespace __detail
- namespace typelist

Classes

- struct __alloc_traits
- struct __common_pool_policy
- class __mt_alloc
- class __mt_alloc_base
- struct __per_type_pool_policy
- class __pool
- class __pool< false >
- class __pool< true >
- class __pool_alloc
- class __pool_alloc_base
- struct __pool_base
- class __rc_string_base
- class __scoped_lock
- class __versa_string
- struct _Caster
- struct _Char_types
- class _ExtPtr_allocator
- struct _Invalid_type

- class _Pointer_adapter
- class _Relative_pointer_impl
- class _Relative_pointer_impl< const _Tp >
- class _Std_pointer_impl
- class _Temporary_buffer
- struct _Unqualified_type
- class allocator
- struct annotate_base
- class binary_compose
- class bitmap_allocator
- struct char_traits
- struct character
- struct condition_base
- struct constant_binary_fun
- struct constant_unary_fun
- struct constant_void_fun
- class debug_allocator
- class enc_filebuf
- struct encoding_char_traits
- class encoding_state
- struct equal_to
- struct forced_error
- class free_list
- class hash_map
- class hash_multimap
- class hash_multiset
- class hash_set
- struct limit_condition
- class malloc_allocator
- class new_allocator
- struct pair
- struct project1st
- struct project2nd
- struct random_condition
- struct rb_tree
- class recursive_init_error
- class rope
- struct select1st
- struct select2nd
- class slist
- class stdio_filebuf
- class stdio_sync_filebuf
- class subtractive_rng
- struct temporary_buffer
- class throw_allocator_base
- struct throw_allocator_limit
- struct throw_allocator_random
- struct throw_value_base
- struct throw_value_limit
- struct throw_value_random
- class unary_compose

Typedefs

- typedef void(∗ **__destroy_handler**) (void ∗)
- template<typename _Tp>
  using __int_traits
- typedef __versa_string< char, std::char_traits< char >, std::allocator< char >, __rc_string_base > **__rc_string**
- typedef __vstring **__sso_string**
- typedef __versa_string< char16_t, std::char_traits< char16_t >, std::allocator< char16_t >, __rc_string_base > **__u16rc_string**
- typedef __u16vstring **__u16sso_string**
- typedef __versa_string< char16_t > **__u16vstring**
- typedef __versa_string< char32_t, std::char_traits< char32_t >, std::allocator< char32_t >, __rc_string_base > **__u32rc_string**
- typedef __u32vstring **__u32sso_string**
- typedef __versa_string< char32_t > **__u32vstring**
- typedef __versa_string< char > **__vstring**
- typedef __versa_string< wchar_t, std::char_traits< wchar_t >, std::allocator< wchar_t >, __rc_string_base > **__wrc_string**
- typedef __wvstring **__wsso_string**
- typedef __versa_string< wchar_t > **__wvstring**
- typedef rope< char > **crope**
- typedef __SIZE_TYPE__ **size_t**
- typedef rope< wchar_t > **wrope**

Enumerations

- enum { **_S_num_primes** }
- enum **_Lock_policy** { **_S_single** , **_S_mutex** , **_S_atomic** }

Functions

- void **__atomic_add** (volatile _Atomic_word ∗__mem, int __val)
- void **__atomic_add_dispatch** (_Atomic_word ∗__mem, int __val)
- void **__atomic_add_single** (_Atomic_word ∗__mem, int __val)
- template<class _Tp>
  void **__aux_require_boolean_expr** (const _Tp &__t)
- template<typename _ToType, typename _FromType>
  _ToType **__const_pointer_cast** (_FromType ∗__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__const_pointer_cast** (const _FromType &__arg)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  std::pair< _InputIterator, _OutputIterator > **__copy_n** (_InputIterator __first, _Size __count, _OutputIterator _↩
  _result, std::input_iterator_tag)
- template<typename _RAIterator, typename _Size, typename _OutputIterator>
  std::pair< _RAIterator, _OutputIterator > **__copy_n** (_RAIterator __first, _Size __count, _OutputIterator __result, std::random_access_iterator_tag)
- template<typename _InputIterator, typename _Distance>
  void **__distance** (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input_iterator_tag)
- template<typename _RandomAccessIterator, typename _Distance>
  void **__distance** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random_access_iterator_tag
- template<typename _ToType, typename _FromType>
  _ToType **__dynamic_pointer_cast** (_FromType ∗__arg)

- template<typename _ToType, typename _FromType>
  _ToType **__dynamic_pointer_cast** (const _FromType &__arg)
- void **__error_type_must_be_a_signed_integer_type** ()
- void **__error_type_must_be_an_integer_type** ()
- void **__error_type_must_be_an_unsigned_integer_type** ()
- _Atomic_word **__exchange_and_add** (volatile _Atomic_word *__mem, int __val)
- _Atomic_word **__exchange_and_add_dispatch** (_Atomic_word *__mem, int __val)
- _Atomic_word **__exchange_and_add_single** (_Atomic_word *__mem, int __val)
- template<class _Concept>
  constexpr void **__function_requires** ()
- template<typename _Type>
  constexpr bool **__is_null_pointer** (_Type *__ptr)
- template<typename _Type>
  constexpr bool **__is_null_pointer** (_Type)
- constexpr bool **__is_null_pointer** (std::nullptr_t)
- bool **__is_single_threaded** () noexcept
- template<typename _InputIterator1, typename _InputIterator2>
  int **__lexicographical_compare_3way** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- int **__lexicographical_compare_3way** (const char *__first1, const char *__last1, const char *__first2, const char *__last2)
- int **__lexicographical_compare_3way** (const unsigned char *__first1, const unsigned char *__last1, const unsigned char *__first2, const unsigned char *__last2)
- template<typename _Tp>
  const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- template<typename _Tp, typename _Compare>
  const _Tp & __median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- crope::reference **__mutable_reference_at** (crope &__c, std::size_t __i)
- template<typename _Tp, typename _Integer>
  _Tp **__power** (_Tp __x, _Integer __n)
- template<typename _Tp, typename _Integer, typename _MonoidOperation>
  _Tp **__power** (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance>
  _RandomAccessIterator **__random_sample** (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance>
  _RandomAccessIterator **__random_sample** (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out, const _Distance __n)
- template<typename _ToType, typename _FromType>
  _ToType **__reinterpret_pointer_cast** (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__reinterpret_pointer_cast** (const _FromType &__arg)
- _Slist_node_base * **__slist_make_link** (_Slist_node_base *__prev_node, _Slist_node_base *__new_node)
- _Slist_node_base * **__slist_previous** (_Slist_node_base *__head, const _Slist_node_base *__node)
- const _Slist_node_base * **__slist_previous** (const _Slist_node_base *__head, const _Slist_node_base *__node)
- _Slist_node_base * **__slist_reverse** (_Slist_node_base *__node)
- std::size_t **__slist_size** (_Slist_node_base *__node)
- void **__slist_splice_after** (_Slist_node_base *__pos, _Slist_node_base *__before_first, _Slist_node_base *__before_last)
- void **__slist_splice_after** (_Slist_node_base *__pos, _Slist_node_base *__head)

- template<typename _ToType, typename _FromType>
  _ToType __static_pointer_cast (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType __static_pointer_cast (const _FromType &__arg)
- size_t **__stl_hash_string** (const char *__s)
- unsigned long **__stl_next_prime** (unsigned long __n)
- template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>
  _Ret **__stoa** (_TRet(*__convf)(const _CharT *, _CharT **, _Base...), const char *__name, const _CharT *__str, std::size_t *__idx, _Base... __base)
- void **__throw_concurrence_lock_error** ()
- void **__throw_concurrence_unlock_error** ()
- void **__throw_forced_error** ()
- template<typename _String, typename _CharT = typename _String::value_type>
  _String **__to_xstring** (int(*__convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n, const _CharT *__fmt,...)
- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > **__uninitialized_copy_n** (_InputIter __first, _Size __count, _ForwardIter __result)
- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > **__uninitialized_copy_n** (_InputIter __first, _Size __count, _ForwardIter __result, std::input_iterator_tag)
- template<typename _RandomAccessIter, typename _Size, typename _ForwardIter>
  std::pair< _RandomAccessIter, _ForwardIter > **__uninitialized_copy_n** (_RandomAccessIter __first, _Size _↩
  _count, _ForwardIter __result, std::random_access_iterator_tag)
- template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator>
  std::pair< _InputIter, _ForwardIter > **__uninitialized_copy_n_a** (_InputIter __first, _Size __count, _ForwardIter __result, _Allocator __alloc)
- template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp>
  std::pair< _InputIter, _ForwardIter > **__uninitialized_copy_n_a** (_InputIter __first, _Size __count, _ForwardIter __result, std::allocator< _Tp >)
- void __verbose_terminate_handler ()
- std::size_t _Bit_scan_forward (std::size_t __num)
- template<typename _ForwardIterator, typename _Allocator>
  void **_Destroy_const** (_ForwardIterator __first, _ForwardIterator __last, _Allocator __alloc)
- template<typename _ForwardIterator, typename _Tp>
  void **_Destroy_const** (_ForwardIterator __first, _ForwardIterator __last, std::allocator< _Tp >)
- template<class _CharT, class _Traits>
  void **_Rope_fill** (std::basic_ostream< _CharT, _Traits > &__o, std::size_t __n)
- template<class _CharT>
  bool **_Rope_is_simple** (_CharT *)
- bool **_Rope_is_simple** (char *)
- bool **_Rope_is_simple** (wchar_t *)
- template<class _Rope_iterator>
  void **_Rope_rotate** (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)
- template<class _CharT>
  void **_S_cond_store_eos** (_CharT &)
- void **_S_cond_store_eos** (char &__c)
- void **_S_cond_store_eos** (wchar_t &__c)
- template<class _CharT>
  _CharT **_S_eos** (_CharT *)
- template<class _CharT>
  bool **_S_is_basic_char_type** (_CharT *)

- bool **_S_is_basic_char_type** (char *)
- bool **_S_is_basic_char_type** (wchar_t *)
- template<class _CharT>
  bool **_S_is_one_byte_char_type** (_CharT *)
- bool **_S_is_one_byte_char_type** (char *)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type airy_ai (_Tp __x)
- float airy_aif (float __x)
- long double airy_ail (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type airy_bi (_Tp __x)
- float airy_bif (float __x)
- long double airy_bil (long double __x)
- template<class _Operation1, class _Operation2>
  unary_compose< _Operation1, _Operation2 > compose1 (const _Operation1 &__fn1, const _Operation2 &_↩
  _fn2)
- template<class _Operation1, class _Operation2, class _Operation3>
  binary_compose< _Operation1, _Operation2, _Operation3 > compose2 (const _Operation1 &__fn1, const _↩
  Operation2 &__fn2, const _Operation3 &__fn3)
- template<typename _Tpa, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)
- float conf_hypergf (float __a, float __c, float __x)
- long double conf_hypergl (long double __a, long double __c, long double __x)
- template<class _Result>
  constant_void_fun< _Result > constant0 (const _Result &__val)
- template<class _Result>
  constant_unary_fun< _Result, _Result > constant1 (const _Result &__val)
- template<class _Result>
  constant_binary_fun< _Result, _Result, _Result > constant2 (const _Result &__val)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  std::pair< _InputIterator, _OutputIterator > copy_n (_InputIterator __first, _Size __count, _OutputIterator __↩
  result)
- template<typename _InputIterator, typename _Tp, typename _Size>
  void **count** (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)
- template<typename _InputIterator, typename _Predicate, typename _Size>
  void **count_if** (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)
- template<typename _InputIterator, typename _Distance>
  void distance (_InputIterator __first, _InputIterator __last, _Distance &__n)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float hypergf (float __a, float __b, float __c, float __x)
- long double hypergl (long double __a, long double __b, long double __c, long double __x)
- template<class _Tp>
  _Tp identity_element (std::multiplies< _Tp >)
- template<class _Tp>
  _Tp identity_element (std::plus< _Tp >)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)
- template<typename _RAIter>
  constexpr bool **is_heap** (_RAIter, _RAIter)
- template<typename _FIter>
  constexpr bool **is_sorted** (_FIter, _FIter)

- template<typename _InputIterator1, typename _InputIterator2>
  int lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<class _Ret, class _Tp, class _Arg>
  std::const_mem_fun1_t< _Ret, _Tp, _Arg > **mem_fun1** (_Ret(_Tp::*__f)(_Arg) const)
- template<class _Ret, class _Tp, class _Arg>
  std::mem_fun1_t< _Ret, _Tp, _Arg > **mem_fun1** (_Ret(_Tp::*__f)(_Arg))
- template<class _Ret, class _Tp, class _Arg>
  std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > **mem_fun1_ref** (_Ret(_Tp::*__f)(_Arg) const)
- template<class _Ret, class _Tp, class _Arg>
  std::mem_fun1_ref_t< _Ret, _Tp, _Arg > **mem_fun1_ref** (_Ret(_Tp::*__f)(_Arg))
- template<typename _Tp1, typename _Tp2>
  bool **operator!=** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator!= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp>
  bool **operator!=** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp>
  bool **operator!=** (const _Pointer_adapter< _Tp > &__lhs, int __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator!=** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator!=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<class _CharT, class _Alloc>
  bool **operator!=** (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator!=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator!=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  bool **operator!=** (const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>
  bool **operator!=** (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)
- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  bool **operator!=** (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>
  bool **operator!=** (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>
  bool **operator!=** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<class _CharT, class _Alloc>
  bool **operator!=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)

- template<class _Tp, class _Alloc>
  bool **operator!=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _Tp>
  bool **operator!=** (int __lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Cond>
  throw_value_base< _Cond > **operator*** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (__versa_string< _CharT, _Traits, _Alloc, _Base > &&__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (_CharT __lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > **operator+** (const _CharT *__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<class _CharT, class _Alloc>
  _Rope_const_iterator< _CharT, _Alloc > **operator+** (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)
- template<class _CharT, class _Alloc>
  _Rope_iterator< _CharT, _Alloc > **operator+** (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **operator+** (const rope< _CharT, _Alloc > &__left, _CharT __right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **operator+** (const rope< _CharT, _Alloc > &__left, const _CharT *__right)

- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **operator+** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &_↩
  _right)
- template<typename _Cond>
  throw_value_base< _Cond > **operator+** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<class _CharT, class _Alloc>
  _Rope_const_iterator< _CharT, _Alloc > **operator+** (std::ptrdiff_t __n, const _Rope_const_iterator< _CharT, _Alloc > &__x)
- template<class _CharT, class _Alloc>
  _Rope_iterator< _CharT, _Alloc > **operator+** (std::ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > & **operator+=** (rope< _CharT, _Alloc > &__left, _CharT __right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > & **operator+=** (rope< _CharT, _Alloc > &__left, const _CharT *__right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > & **operator+=** (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT, class _Alloc>
  std::ptrdiff_t **operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _↩
  CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  _Rope_const_iterator< _CharT, _Alloc > **operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__↩
  x, std::ptrdiff_t __n)
- template<class _CharT, class _Alloc>
  std::ptrdiff_t **operator-** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  _Rope_iterator< _CharT, _Alloc > **operator-** (const _Rope_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)
- template<typename _Cond>
  throw_value_base< _Cond > **operator-** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Tp1, typename _Tp2>
  bool **operator**< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator< (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<class _CharT, class _Alloc>
  bool **operator**< (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator**< (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<typename _Value, typename _Int, typename _St>
  bool **operator**< (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)
- template<class _CharT, class _Alloc>
  bool **operator**< (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)

- template<class _Tp, class _Alloc>
  bool **operator**< (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _Cond>
  bool **operator**< (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<class _CharT, class _Traits, class _Alloc>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__o, const rope< _CharT, _Alloc > &__r)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::beta_distribution< _RealType > &__x)
- template<typename _UIntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)
- template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)
- template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const ↩ __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::triangular_distribution< _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::von_mises_distribution< _RealType > &__x)
- template<typename _CharT, typename _Traits, typename _StoreT>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const _Pointer_adapter< _StoreT > &__p)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const arcsine_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const hoyt_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const k_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const logistic_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const nakagami_distribution< _RealType > &__x)

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const pareto_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const rice_distribution< _RealType > &__x)
- std::ostream & **operator**<< (std::ostream &os, const annotate_base &__b)
- template<typename _Tp1, typename _Tp2>
  bool **operator**<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp>
  bool **operator**<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**<= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<class _CharT, class _Alloc>
  bool **operator**<= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator**<= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator**<= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _Tp, class _Alloc>
  bool **operator**<= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _Tp1, typename _Tp2>
  bool **operator==** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<size_t _Dimen, typename _RealType>
  bool **operator==** (const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d1, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d2)
- template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>
  bool **operator==** (const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__lhs, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__rhs)
- template<typename _Tp, typename _Poolp>
  bool **operator==** (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)
- template<typename _Tp>
  bool **operator==** (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator== (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp>
  bool operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp>
  bool **operator==** (const _Pointer_adapter< _Tp > &__lhs, int __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator==** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator==** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<class _CharT, class _Alloc>
  bool **operator==** (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_proxy< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator==** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator==** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<typename _Tp1, typename _Tp2>
  bool **operator==** (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()
- template<typename _Value, typename _Int, typename _St>
  bool **operator==** (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  bool **operator==** (const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>
  bool **operator==** (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)
- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  bool **operator==** (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>
  bool **operator==** (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>
  bool **operator==** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<class _CharT, class _Alloc>
  bool **operator==** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _Tp, class _Alloc>
  bool **operator==** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _Tp, typename _Cond>
  bool **operator==** (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- template<typename _Cond>
  bool **operator==** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Tp>
  bool **operator==** (int __lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**> (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator> (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp>
  bool **operator**> (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**> (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**> (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<class _CharT, class _Alloc>
  bool **operator**> (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator**> (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator**> (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _Tp, class _Alloc>
  bool **operator**> (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _Tp1, typename _Tp2>
  bool **operator**>= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp>
  bool **operator**>= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**>= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **operator**>= (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<class _CharT, class _Alloc>
  bool **operator**>= (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator**>= (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **operator**>= (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _Tp, class _Alloc>
  bool **operator**>= (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::beta_distribution< _RealType > &__x)
- template<typename _UIntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)
- template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)

- template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::triangular_distribution< _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, __gnu_cxx::von_mises_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, arcsine_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, hoyt_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, k_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, logistic_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, nakagami_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, pareto_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, rice_distribution< _RealType > &__x)
- template<typename _Tp, typename _Integer>
  _Tp power (_Tp __x, _Integer __n)
- template<typename _Tp, typename _Integer, typename _MonoidOperation>
  _Tp power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _InputIterator, typename _RandomAccessIterator>
  _RandomAccessIterator random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>
  _RandomAccessIterator random_sample (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>
  _OutputIterator random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)

- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator>
  _OutputIterator random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)
- void **rotate** (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_iterator< char, __↩
  STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __last)
- double **stod** (const __vstring &__str, std::size_t *__idx=0)
- double **stod** (const __wvstring &__str, std::size_t *__idx=0)
- float **stof** (const __vstring &__str, std::size_t *__idx=0)
- float **stof** (const __wvstring &__str, std::size_t *__idx=0)
- int **stoi** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- int **stoi** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- long **stol** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- long **stol** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- long double **stold** (const __vstring &__str, std::size_t *__idx=0)
- long double **stold** (const __wvstring &__str, std::size_t *__idx=0)
- long long **stoll** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- long long **stoll** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- unsigned long **stoul** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- unsigned long **stoul** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- unsigned long long **stoull** (const __vstring &__str, std::size_t *__idx, int __base=10)
- unsigned long long **stoull** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  void swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _Tp>
  void **swap** (_ExtPtr_allocator< _Tp > &__larg, _ExtPtr_allocator< _Tp > &__rarg)
- template<class _CharT, class __Alloc>
  void **swap** (_Rope_char_ref_proxy< _CharT, __Alloc > __a, _Rope_char_ref_proxy< _CharT, __Alloc > __b)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  void **swap** (hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  void **swap** (hash_multimap< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  void **swap** (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  void **swap** (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All>
  void **swap** (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)
- template<class _CharT, class _Alloc>
  void **swap** (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)
- template<class _Tp, class _Alloc>
  void **swap** (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)
- template<typename _Cond>
  void **swap** (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)
- __vstring **to_string** (double __val)
- __vstring **to_string** (float __val)

- __vstring **to_string** (int __val)
- __vstring **to_string** (long __val)
- __vstring **to_string** (long double __val)
- __vstring **to_string** (long long __val)
- __vstring **to_string** (unsigned __val)
- __vstring **to_string** (unsigned long __val)
- __vstring **to_string** (unsigned long long __val)
- __wvstring **to_wstring** (double __val)
- __wvstring **to_wstring** (float __val)
- __wvstring **to_wstring** (int __val)
- __wvstring **to_wstring** (long __val)
- __wvstring **to_wstring** (long double __val)
- __wvstring **to_wstring** (long long __val)
- __wvstring **to_wstring** (unsigned __val)
- __wvstring **to_wstring** (unsigned long __val)
- __wvstring **to_wstring** (unsigned long long __val)
- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > uninitialized_copy_n (_InputIter __first, _Size __count, _ForwardIter __result)

Variables

- const _Lock_policy **__default_lock_policy**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__e**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__gamma_e**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__ln_10**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__ln_2**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__ln_3**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__log10_e**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__log2_e**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__one_div_e**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__one_div_pi**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__one_div_root_2**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__phi**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__pi**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__pi_half**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__pi_quarter**

- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__pi_third**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__root_2**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__root_3**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__root_5**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__root_7**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__root_pi_div_2**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__two_div_pi**
- template<typename _RealType>
  constexpr _RealType **__math_constants**< **_RealType** >**::__two_div_root_pi**
- template<typename _Value>
  const int **__numeric_traits_floating**< **_Value** >**::__digits10**
- template<typename _Value>
  const bool **__numeric_traits_floating**< **_Value** >**::__is_signed**
- template<typename _Value>
  const int **__numeric_traits_floating**< **_Value** >**::__max_digits10**
- template<typename _Value>
  const int **__numeric_traits_floating**< **_Value** >**::__max_exponent10**
- template<typename _Value>
  const int **__numeric_traits_integer**< **_Value** >**::__digits**
- template<typename _Value>
  const bool **__numeric_traits_integer**< **_Value** >**::__is_signed**
- template<typename _Value>
  const _Value **__numeric_traits_integer**< **_Value** >**::__max**
- template<typename _Value>
  const _Value **__numeric_traits_integer**< **_Value** >**::__min**
- template<typename _Tp>
  _Atomic_word **__pool_alloc**< **_Tp** >**::_S_force_new**
- template<typename _CharT, typename _Traits, typename _Alloc>
  __rc_string_base< _CharT, _Traits, _Alloc >::_Rep_empty **__rc_string_base**< **_CharT, _Traits, _Alloc** >**::_S_empty_rep**
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  const __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type **__versa_string**< **_CharT, _Traits, _Alloc, _Base** >**::npos**
- template<typename _PrimeType>
  const _PrimeType **_Hashtable_prime_list**< **_PrimeType** >**::__stl_prime_list** [_S_num_primes]
- template<typename _Tp>
  std::size_t **bitmap_allocator**< **_Tp** >**::_S_block_size**
- template<typename _Tp>
  bitmap_allocator< _Tp >::_BPVector::size_type **bitmap_allocator**< **_Tp** >**::_S_last_dealloc_index**
- template<typename _Tp>
  bitmap_allocator< _Tp >::_BPVector **bitmap_allocator**< **_Tp** >**::_S_mem_blocks**
- template<typename _Tp>
  bitmap_allocator< _Tp >::__mutex_type **bitmap_allocator**< **_Tp** >**::_S_mut**
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **identity_element** (_Rope_Concat_fn< _CharT, _Alloc >)

- template<class _CharT, class _Alloc>
  _CharT **rope< _CharT, _Alloc >::_S_empty_c_str** [1]
- template<class _CharT, class _Alloc>
  const unsigned long **rope< _CharT, _Alloc >::_S_min_len** [int(__detail::_S_max_rope_depth)+1]
- template<class _CharT, class _Alloc>
  const rope< _CharT, _Alloc >::size_type **rope< _CharT, _Alloc >::npos**

4.1.1 Detailed Description

GNU extensions for public use.

4.1.2 Typedef Documentation

__int_traits

```
template<typename _Tp>
using __gnu_cxx::__int_traits
```

Convenience alias for __numeric_traits<integer-type>.

4.1.3 Function Documentation

__static_pointer_cast() [1/2]

```
template<typename _ToType, typename _FromType>
_ToType __gnu_cxx::__static_pointer_cast (
            _FromType * __arg)  [inline]
```

Casting operations for cases where _FromType is a standard pointer. _ToType can be a standard or non-standard pointer.

__static_pointer_cast() [2/2]

```
template<typename _ToType, typename _FromType>
_ToType __gnu_cxx::__static_pointer_cast (
            const _FromType & __arg)  [inline]
```

Casting operations for cases where _FromType is not a standard pointer. _ToType can be a standard or non-standard pointer. Given that _FromType is not a pointer, it must have a get() method that returns the standard pointer equivalent of the address it points to, and must have an element_type typedef which names the type it points to.

_Bit_scan_forward()

```
std::size_t __gnu_cxx::_Bit_scan_forward (
            std::size_t __num)  [inline]
```

Generic Version of the bsf instruction.
Referenced by __gnu_cxx::bitmap_allocator< _Tp1 >::_M_allocate_single_object().

iota()

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

Create a range of sequentially increasing values.
For each element in the range [first,last) assigns `value` and increments `value` as if by `++value`.

Parameters

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

operator"!=() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator!= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test difference of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs.compare*(*__rhs*) != 0. False otherwise.

operator"!=() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator!= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)  [inline]
```

Test difference of string and C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs.compare*(*__rhs*) != 0. False otherwise.

operator"!=() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator!= (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test difference of C string and string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__rhs.compare*(*__lhs*) != 0. False otherwise.

operator+() [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
            _CharT __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate character and string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Allo
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Bas

operator+() [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            _CharT __rhs)
```

Concatenate string and character.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Allo
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve(), and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Bas

operator+() [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate two strings.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with value of *__lhs* followed by *__rhs*.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Allo
and __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

operator+() [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)
```

Concatenate string and C string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

operator+() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)
```

Concatenate C string and string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with value of *__lhs* followed by *__rhs*.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size().

operator<() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator< (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if string precedes string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator<() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator< (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)  [inline]
```

Test if string precedes C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator<() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator< (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if C string precedes string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* precedes *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator<=() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator<= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if string doesn't follow string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* doesn't follow *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator<=() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator<= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)  [inline]
```

Test if string doesn't follow C string.

Parameters

__lhs	String.

__rhs	C string.

Returns

True if *__lhs* doesn't follow *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator<=() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator<= (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if C string doesn't follow string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* doesn't follow *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator==() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator== (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test equivalence of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs.compare*(*__rhs*) == 0. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data(), and __gnu_cxx::__versa_string< _CharT, _Traits, _All

operator==() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
bool __gnu_cxx::operator== (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)  [inline]
```

Test equivalence of string and C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs.compare*(*__rhs*) == 0. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data(), and __gnu_cxx::__versa_string< _CharT, _Traits, _All

operator==() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator== (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test equivalence of C string and string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__rhs.compare*(*__lhs*) == 0. False otherwise.

operator==() [4/4]

```
template<typename _Tp>
bool __gnu_cxx::operator== (
            const _Pointer_adapter< _Tp > & __lhs,
            const _Pointer_adapter< _Tp > & __rhs)  [inline]
```

Comparison operators for _Pointer_adapter defer to the base class' comparison operators, when possible.

operator>() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator> (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if string follows string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* follows *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator>() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator> (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)  [inline]
```

Test if string follows C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* follows *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator>() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator> (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if C string follows string.

Parameters

__lhs	C string.
__rhs	String.

Returns

True if *__lhs* follows *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator>=() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator>= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if string doesn't precede string.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs* doesn't precede *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator>=() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator>= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
            const _CharT * __rhs)  [inline]
```

Test if string doesn't precede C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs* doesn't precede *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

operator>=() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::operator>= (
            const _CharT * __lhs,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Test if C string doesn't precede string.

Parameters

__lhs	C string.

__rhs	String.

Returns

True if *__lhs* doesn't precede *__rhs*. False otherwise.

References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare().

swap()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::swap (
             __versa_string< _CharT, _Traits, _Alloc, _Base > & __lhs,
             __versa_string< _CharT, _Traits, _Alloc, _Base > & __rhs)  [inline]
```

Swap contents of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Exchanges the contents of *__lhs* and *__rhs* in constant time.
References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap().

4.2 __gnu_cxx::__detail Namespace Reference

Classes

- class __mini_vector
- class _Bitmap_counter
- class _Ffit_finder

Enumerations

- enum { **_S_max_rope_depth** }
- enum { **bits_per_byte** , **bits_per_block** }
- enum **_Tag** { **_S_leaf** , **_S_concat** , **_S_substringfn** , **_S_function** }

Functions

- void __bit_allocate (std::size_t *__pbmap, std::size_t __pos) throw ()
- void __bit_free (std::size_t *__pbmap, std::size_t __pos) throw ()
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  _ForwardIterator **__lower_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _AddrPair>
  std::size_t __num_bitmaps (_AddrPair __ap)
- template<typename _AddrPair>
  std::size_t __num_blocks (_AddrPair __ap)

4.2.1 Detailed Description

Implementation details not part of the namespace __gnu_cxx interface.

4.2.2 Function Documentation

__bit_allocate()

```
void __gnu_cxx::__detail::__bit_allocate (
             std::size_t * __pbmap,
             std::size_t __pos) throw ( )   [inline]
```

Mark a memory address as allocated by re-setting the corresponding bit in the bit-map.
Referenced by __gnu_cxx::bitmap_allocator< _Tp1 >::_M_allocate_single_object().

__bit_free()

```
void __gnu_cxx::__detail::__bit_free (
             std::size_t * __pbmap,
             std::size_t __pos) throw ( )   [inline]
```

Mark a memory address as free by setting the corresponding bit in the bit-map.
Referenced by __gnu_cxx::bitmap_allocator< _Tp1 >::_M_deallocate_single_object().

__num_bitmaps()

```
template<typename _AddrPair>
std::size_t __gnu_cxx::__detail::__num_bitmaps (
             _AddrPair __ap)  [inline]
```

The number of Bit-maps pointed to by the address pair passed to the function.
References __num_blocks().
Referenced by __gnu_cxx::bitmap_allocator< _Tp1 >::_M_allocate_single_object(), and __gnu_cxx::bitmap_allocator< _Tp1 >::_M_dea

__num_blocks()

```
template<typename _AddrPair>
std::size_t __gnu_cxx::__detail::__num_blocks (
             _AddrPair __ap)  [inline]
```

The number of Blocks pointed to by the address pair passed to the function.
Referenced by __num_bitmaps().

4.3 __gnu_cxx::typelist Namespace Reference

Functions

- template<typename Fn, typename Typelist>
  void **apply** (Fn &, Typelist)
- template<typename Fn, typename Typelist>
  void **apply_generator** (Fn &fn, Typelist)
- template<typename Fn, typename TypelistT, typename TypelistV>
  void **apply_generator** (Fn &fn, TypelistT, TypelistV)
- template<typename Gn, typename Typelist>
  void apply_generator (Gn &, Typelist)
- template<typename Gn, typename TypelistT, typename TypelistV>
  void **apply_generator** (Gn &, TypelistT, TypelistV)

4.3.1 Detailed Description

GNU typelist extensions for public compile-time use.

4.3.2 Function Documentation

apply_generator()

```
template<typename Gn, typename Typelist>
void __gnu_cxx::typelist::apply_generator (
            Gn & ,
            Typelist )
```

Apply all typelist types to generator functor.

4.4 __gnu_debug Namespace Reference

Classes

- class _After_nth_from
- struct _BeforeBeginHelper
- class _Equal_to
- class _Not_equal_to
- class _Safe_container
- class _Safe_forward_list
- class _Safe_iterator
- class _Safe_iterator_base
- class _Safe_local_iterator
- class _Safe_local_iterator_base
- class _Safe_node_sequence
- class _Safe_sequence
- class _Safe_sequence_base
- class _Safe_unordered_container
- class _Safe_unordered_container_base
- class _Safe_vector
- struct _Sequence_traits
- class basic_string
- class type_info

Typedefs

- typedef basic_string< char > **string**
- typedef basic_string< char16_t > u16string
- typedef basic_string< char32_t > u32string
- typedef basic_string< wchar_t > **wstring**

Enumerations

- enum **_Debug_msg_id** {
  __msg_valid_range , **__msg_insert_singular** , **__msg_insert_different** , **__msg_erase_bad** ,
  __msg_erase_different , **__msg_subscript_oob** , **__msg_empty** , **__msg_unpartitioned** ,
  __msg_unpartitioned_pred , **__msg_unsorted** , **__msg_unsorted_pred** , **__msg_not_heap** ,
  __msg_not_heap_pred , **__msg_bad_bitset_write** , **__msg_bad_bitset_read** , **__msg_bad_bitset_flip** ,
  __msg_self_splice , **__msg_splice_alloc** , **__msg_splice_bad** , **__msg_splice_other** ,
  __msg_splice_overlap , **__msg_init_singular** , **__msg_init_copy_singular** , **__msg_init_const_singular** ,
  __msg_copy_singular , **__msg_bad_deref** , **__msg_bad_inc** , **__msg_bad_dec** ,
  __msg_iter_subscript_oob , **__msg_advance_oob** , **__msg_retreat_oob** , **__msg_iter_compare_bad** ,
  __msg_compare_different , **__msg_iter_order_bad** , **__msg_order_different** , **__msg_distance_bad** ,
  __msg_distance_different , **__msg_deref_istream** , **__msg_inc_istream** , **__msg_output_ostream** ,

__msg_deref_istreambuf , **__msg_inc_istreambuf** , **__msg_insert_after_end** , **__msg_erase_after_bad** , **__msg_valid_range2** , **__msg_local_iter_compare_bad** , **__msg_non_empty_range** , **__msg_self_move_assign** ,
__msg_bucket_index_oob , **__msg_valid_load_factor** , **__msg_equal_allocs** , **__msg_insert_range_from_self** ,
__msg_irreflexive_ordering }

- enum _Distance_precision {
__dp_none , **__dp_equality** , **__dp_sign** , **__dp_sign_max_size** ,
__dp_exact }

Functions

- template<typename _Iterator>
constexpr _Iterator __base (_Iterator __it)
- template<typename _Iterator, typename _Sequence>
constexpr _Iterator **__base** (const _Safe_iterator< _Iterator, _Sequence, std::random_access_iterator_tag > &__it)
- template<typename _Iterator>
constexpr auto **__base** (const std::move_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(__base(__it.base())))
- template<typename _Iterator, typename _Sequence>
constexpr std::reverse_iterator< _Iterator > **__base** (const std::reverse_iterator< _Safe_iterator< _Iterator, _Sequence, std::random_access_iterator_tag > > &__it)
- template<typename _InputIterator, typename _Size>
constexpr bool **__can_advance** (_InputIterator, _Size)
- template<typename _InputIterator, typename _Diff>
constexpr bool **__can_advance** (_InputIterator, const std::pair< _Diff, _Distance_precision > &, int)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Size>
constexpr bool **__can_advance** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, _Size)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Diff>
constexpr bool **__can_advance** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const std::pair< _Diff, _Distance_precision > &, int)
- template<typename _Iterator, typename _Size>
constexpr bool **__can_advance** (const std::move_iterator< _Iterator > &__it, _Size __n)
- template<typename _Iterator, typename _Diff>
constexpr bool **__can_advance** (const std::move_iterator< _Iterator > &__it, const std::pair< _Diff, _Distance_precision > &__dist, int __way)
- template<typename _Iterator, typename _Size>
constexpr bool **__can_advance** (const std::reverse_iterator< _Iterator > &__it, _Size __n)
- template<typename _Iterator, typename _Diff>
constexpr bool **__can_advance** (const std::reverse_iterator< _Iterator > &__it, const std::pair< _Diff, _Distance_precision > &__dist, int __way)
- template<typename _ForwardIterator, typename _Tp>
constexpr bool **__check_partitioned_lower** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _ForwardIterator, typename _Tp, typename _Pred>
constexpr bool **__check_partitioned_lower** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)
- template<typename _ForwardIterator, typename _Tp>
constexpr bool **__check_partitioned_upper** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _ForwardIterator, typename _Tp, typename _Pred>
constexpr bool **__check_partitioned_upper** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)

- template<typename _Iterator>
  constexpr bool **__check_singular** (_Iterator const &)
- template<typename _Tp>
  constexpr bool __check_singular (_Tp *const &__ptr)
- bool __check_singular_aux (const _Safe_iterator_base *__x)
- bool **__check_singular_aux** (const class _Safe_iterator_base *)
- bool **__check_singular_aux** (const void *)
- template<typename _InputIterator>
  constexpr bool **__check_sorted** (const _InputIterator &__first, const _InputIterator &__last)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__check_sorted** (const _InputIterator &__first, const _InputIterator &__last, _Predicate __pred)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr bool **__check_sorted_aux** (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, std::forward_iterator_tag)
- template<typename _ForwardIterator>
  constexpr bool **__check_sorted_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__check_sorted_aux** (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)
- template<typename _InputIterator>
  constexpr bool **__check_sorted_aux** (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr bool **__check_sorted_set** (const _InputIterator1 &__first, const _InputIterator1 &__last, const _Input↩
  Iterator2 &)
- template<typename _InputIterator1, typename _InputIterator2, typename _Predicate>
  constexpr bool **__check_sorted_set** (const _InputIterator1 &__first, const _InputIterator1 &__last, const _Input↩
  Iterator2 &, _Predicate __pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__check_sorted_set_aux** (const _InputIterator &, const _InputIterator &, _Predicate, std::__↩
  false_type)
- template<typename _InputIterator>
  constexpr bool **__check_sorted_set_aux** (const _InputIterator &, const _InputIterator &, std::__false_type)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__check_sorted_set_aux** (const _InputIterator &__first, const _InputIterator &__last, _Predicate
  __pred, std::__true_type)
- template<typename _InputIterator>
  constexpr bool **__check_sorted_set_aux** (const _InputIterator &__first, const _InputIterator &__last, std::__↩
  true_type)
- template<typename _CharT, typename _Integer>
  const _CharT * __check_string (const _CharT *__s, _Integer __n, const char *__file, unsigned int __line, const
  char *__function)
- template<typename _CharT>
  const _CharT * __check_string (const _CharT *__s, const char *__file, unsigned int __line, const char *__↩
  function)
- template<typename _InputIterator>
  constexpr _InputIterator **__check_valid_range** (const _InputIterator &__first, const _InputIterator &__last, const
  char *__file, unsigned int __line, const char *__function)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__foreign_iterator** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, _InputIterator __other,
  _InputIterator __other_end)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator, typename _OtherSequence, typename
  _OtherCategory>
  bool __foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator<
  _OtherIterator, _OtherSequence, _OtherCategory > &, const _Safe_iterator< _OtherIterator, _OtherSequence,
  _OtherCategory > &)

- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__foreign_iterator_aux2** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _Input↩ Iterator &__other, const _InputIterator &__other_end)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator>
  bool __foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _Safe_iterator< _OtherIterator, _Sequence, _Category > &__other, const _Safe_iterator< _OtherIterator, _Sequence, _Category > &)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__foreign_iterator_aux3** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _InputIterator &, const _InputIterator &, std::__false_type)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__foreign_iterator_aux3** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _Input↩ Iterator &__other, const _InputIterator &__other_end, std::__true_type)
- template<typename _Iterator, typename _Sequence, typename _Category>
  bool **__foreign_iterator_aux4** (const _Safe_iterator< _Iterator, _Sequence, _Category > &,...)
- template<typename _Iterator, typename _Sequence, typename _Category>
  bool **__foreign_iterator_aux4** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const typename _Sequence::value_type *__other)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__get_distance** (_Iterator __lhs, _Iterator __rhs)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__get_distance** (_Iterator __lhs, _Iterator __rhs, std::input_iterator_tag)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type __get_distance (_Iterator __lhs, _Iterator __rhs, std::random_access_iterator_tag)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__get_distance** (const std::move_iterator< _Iterator > &__first, const std::move_iterator< _Iterator > &__last)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__get_distance** (const std::reverse_iterator< _Iterator > &_↩ _first, const std::reverse_iterator< _Iterator > &__last)
- template<typename _Iterator>
  constexpr bool **__is_irreflexive** (_Iterator __it)
- template<typename _Iterator, typename _Pred>
  constexpr bool **__is_irreflexive_pred** (_Iterator __it, _Pred __pred)
- template<typename _Iterator>
  constexpr _Iterator **__unsafe** (_Iterator __it)
- template<typename _Iterator, typename _Sequence>
  _Iterator **__unsafe** (const _Safe_iterator< _Iterator, _Sequence > &__it)
- template<typename _Iterator, typename _UContainer>
  _Iterator **__unsafe** (const _Safe_local_iterator< _Iterator, _UContainer > &__it)
- template<typename _Iterator>
  constexpr auto **__unsafe** (const std::move_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(↩ __unsafe(__it.base())))
- template<typename _Iterator>
  constexpr auto **__unsafe** (const std::reverse_iterator< _Iterator > &__it) -> decltype(std::__make_reverse_↩ iterator(__unsafe(__it.base())))
- template<typename _InputIterator>
  constexpr bool **__valid_range** (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  constexpr bool **__valid_range** (_InputIterator __first, _InputIterator __last, typename _Distance_traits< _Input↩ Iterator >::__type &__dist)

- template<typename _Iterator, typename _Sequence, typename _Category>
  constexpr bool **__valid_range** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _Iterator, _Sequence, _Category > &)
- template<typename _Iterator, typename _Sequence, typename _Category>
  constexpr bool __valid_range (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _Iterator, _Sequence, _Category > &, typename _Distance_traits< _Iterator >::__type &)
- template<typename _Iterator, typename _Sequence>
  bool **__valid_range** (const _Safe_local_iterator< _Iterator, _Sequence > &, const _Safe_local_iterator< _Iterator, _Sequence > &)
- template<typename _Iterator, typename _Sequence>
  bool **__valid_range** (const _Safe_local_iterator< _Iterator, _Sequence > &, const _Safe_local_iterator< _Iterator, _Sequence > &, typename _Distance_traits< _Iterator >::__type &)
- template<typename _Iterator, typename _UContainer>
  bool **__valid_range** (const _Safe_local_iterator< _Iterator, _UContainer > &__first, const _Safe_local_iterator< _Iterator, _UContainer > &__last)
- template<typename _Iterator, typename _UContainer>
  bool __valid_range (const _Safe_local_iterator< _Iterator, _UContainer > &__first, const _Safe_local_iterator< _Iterator, _UContainer > &__last, typename _Distance_traits< _Iterator >::__type &__dist_info)
- template<typename _Iterator>
  constexpr bool **__valid_range** (const std::move_iterator< _Iterator > &__first, const std::move_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)
- template<typename _Iterator>
  constexpr bool **__valid_range** (const std::reverse_iterator< _Iterator > &__first, const std::reverse_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)
- template<typename _InputIterator>
  constexpr bool **__valid_range_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _InputIterator>
  constexpr bool **__valid_range_aux** (_InputIterator __first, _InputIterator __last, std::random_access_iterator_tag)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_istream< _CharT, _Traits > & **getline** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_istream< _CharT, _Traits > & **getline** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator!=** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **operator+** (_CharT __lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **operator+** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **operator+** (const basic_string< _CharT, _Traits, _Allocator > &_lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **operator+** (const basic_string< _CharT, _Traits, _Allocator > &_lhs, const _CharT *__rhs)

- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **operator+** (const basic_string< _CharT, _Traits, _Allocator > &_↩
  _lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator<** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator<** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator<** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_ostream< _CharT, _Traits > & **operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator<=** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator<=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator<=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator==** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator==** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator==** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator>** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator>** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator>** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator>=** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator>=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **operator>=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT, typename _Traits, typename _Allocator>
  void **swap** (basic_string< _CharT, _Traits, _Allocator > &__lhs, basic_string< _CharT, _Traits, _Allocator > &↩
  __rhs)

4.4.1 Detailed Description

GNU debug classes for public use.

4.4.2 Typedef Documentation

u16string

```
typedef basic_string<char16_t> __gnu_debug::u16string
```
A string of `char16_t`.

u32string

```
typedef basic_string<char32_t> __gnu_debug::u32string
```
A string of `char32_t`.

4.4.3 Enumeration Type Documentation

_Distance_precision

```
enum __gnu_debug::_Distance_precision
```
The precision to which we can calculate the distance between two iterators.

4.4.4 Function Documentation

__base()

```
template<typename _Iterator>
_Iterator __gnu_debug::__base (
            _Iterator __it)  [inline], [constexpr]
```
Helper function to extract base iterator of random access safe iterator in order to reduce performance impact of debug mode. Limited to random access iterator because it is the only category for which it is possible to check for correct iterators order in the __valid_range function thanks to the < operator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_before_dereferenceable().

__check_singular()

```
template<typename _Tp>
bool __gnu_debug::__check_singular (
            _Tp *const & __ptr)  [inline], [constexpr]
```
Non-NULL pointers are nonsingular.

__check_singular_aux()

```
bool __gnu_debug::__check_singular_aux (
            const _Safe_iterator_base * __x)  [inline]
```
Iterators that derive from _Safe_iterator_base can be determined singular or non-singular.
References __gnu_debug::_Safe_iterator_base::_M_singular().

__check_string() [1/2]

```
template<typename _CharT, typename _Integer>
const _CharT * __gnu_debug::__check_string (
            const _CharT * __s,
            _Integer __n,
            const char * __file,
            unsigned int __line,
            const char * __function)  [inline]
```
Checks that __s is non-NULL or __n == 0, and then returns __s.

__check_string() [2/2]

```
template<typename _CharT>
const _CharT * __gnu_debug::__check_string (
            const _CharT * __s,
            const char * __file,
            unsigned int __line,
            const char * __function)  [inline]
```

Checks that __s is non-NULL and then returns __s.

__foreign_iterator_aux2() [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator,
typename _OtherSequence, typename _OtherCategory>
bool __gnu_debug::__foreign_iterator_aux2 (
            const _Safe_iterator< _Iterator, _Sequence, _Category > & ,
            const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > & ,
            const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > & )  [inline]
```

Handle debug iterators from different types of container.

__foreign_iterator_aux2() [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator>
bool __gnu_debug::__foreign_iterator_aux2 (
            const _Safe_iterator< _Iterator, _Sequence, _Category > & __it,
            const _Safe_iterator< _OtherIterator, _Sequence, _Category > & __other,
            const _Safe_iterator< _OtherIterator, _Sequence, _Category > & )  [inline]
```

Handle debug iterators from the same type of container.

__get_distance()

```
template<typename _Iterator>
_Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (
            _Iterator __lhs,
            _Iterator __rhs,
            std::random_access_iterator_tag )  [inline], [constexpr]
```

Determine the distance between two iterators with some known precision.
References std::make_pair().

__valid_range() [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category>
bool __gnu_debug::__valid_range (
            const _Safe_iterator< _Iterator, _Sequence, _Category > & __first,
            const _Safe_iterator< _Iterator, _Sequence, _Category > & __last,
            typename _Distance_traits< _Iterator >::__type & __dist)  [inline], [constexpr]
```

Safe iterators know how to check if they form a valid range.

__valid_range() [2/2]

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::__valid_range (
            const _Safe_local_iterator< _Iterator, _UContainer > & __first,
            const _Safe_local_iterator< _Iterator, _UContainer > & __last,
            typename _Distance_traits< _Iterator >::__type & __dist_info)  [inline]
```

Safe local iterators know how to check if they form a valid range.

4.5 __gnu_internal Namespace Reference

4.5.1 Detailed Description

GNU implementation details, not for public use or export. Used only when anonymous namespaces cannot be substituted.

4.6 __gnu_parallel Namespace Reference

Classes

- struct __accumulate_binop_reduct
- struct __accumulate_selector
- struct __adjacent_difference_selector
- struct __adjacent_find_selector
- class __binder1st
- class __binder2nd
- struct __count_if_selector
- struct __count_selector
- struct __fill_selector
- struct __find_first_of_selector
- struct __find_if_selector
- struct __for_each_selector
- struct __generate_selector
- struct __generic_find_selector
- struct __generic_for_each_selector
- struct __identity_selector
- struct __inner_product_selector
- struct __max_element_reduct
- struct __min_element_reduct
- struct __mismatch_selector
- struct __multiway_merge_3_variant_sentinel_switch
- struct __multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- struct __multiway_merge_4_variant_sentinel_switch
- struct __multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- struct __multiway_merge_k_variant_sentinel_switch
- struct __multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- struct __replace_if_selector
- struct __replace_selector
- struct __transform1_selector
- struct __transform2_selector
- class __unary_negate
- struct _DRandomShufflingGlobalData
- struct _DRSSorterPU
- struct _DummyReduct
- class _EqualFromLess
- struct _EqualTo
- class _GuardedIterator
- class _IteratorPair
- class _IteratorTriple
- struct _Job
- struct _Less

- class _Lexicographic
- class _LexicographicReverse
- class _LoserTree
- class _LoserTree< false, _Tp, _Compare >
- class _LoserTreeBase
- class _LoserTreePointer
- class _LoserTreePointer< false, _Tp, _Compare >
- class _LoserTreePointerBase
- class _LoserTreePointerUnguarded
- class _LoserTreePointerUnguarded< false, _Tp, _Compare >
- class _LoserTreePointerUnguardedBase
- struct _LoserTreeTraits
- class _LoserTreeUnguarded
- class _LoserTreeUnguarded< false, _Tp, _Compare >
- class _LoserTreeUnguardedBase
- struct _Multiplies
- struct _Nothing
- struct _Piece
- struct _Plus
- struct _PMWMSSortingData
- class _PseudoSequence
- class _PseudoSequenceIterator
- struct _QSBThreadLocal
- class _RandomNumber
- class _RestrictedBoundedConcurrentQueue
- struct _SamplingSorter
- struct _SamplingSorter< false, _RAIter, _StrictWeakOrdering >
- struct _Settings
- struct _SplitConsistently
- struct _SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >
- struct _SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >
- struct balanced_quicksort_tag
- struct balanced_tag
- struct constant_size_blocks_tag
- struct default_parallel_tag
- struct equal_split_tag
- struct exact_tag
- struct find_tag
- struct growing_blocks_tag
- struct multiway_mergesort_exact_tag
- struct multiway_mergesort_sampling_tag
- struct multiway_mergesort_tag
- struct omp_loop_static_tag
- struct omp_loop_tag
- struct parallel_tag
- struct quicksort_tag
- struct sampling_tag
- struct sequential_tag
- struct unbalanced_tag

Typedefs

- typedef unsigned short _BinIndex
- typedef int64_t _CASable
- typedef uint64_t _SequenceIndex
- typedef uint16_t _ThreadIndex

Enumerations

- enum _AlgorithmStrategy { **heuristic** , **force_sequential** , **force_parallel** }
- enum _FindAlgorithm { **GROWING_BLOCKS** , **CONSTANT_SIZE_BLOCKS** , **EQUAL_SPLIT** }
- enum _MultiwayMergeAlgorithm { **LOSER_TREE** }
- enum _Parallelism {
  sequential , parallel_unbalanced , parallel_balanced , parallel_omp_loop ,
  parallel_omp_loop_static , parallel_taskqueue }
- enum _PartialSumAlgorithm { **RECURSIVE** , **LINEAR** }
- enum _SortAlgorithm { **MWMS** , **QS** , **QS_BALANCED** }
- enum _SplittingAlgorithm { **SAMPLING** , **EXACT** }

Functions

- template<typename _Tp>
  _Tp **__add_omp** (volatile _Tp *__ptr, _Tp __addend)
- template<typename _RAIter, typename _DifferenceTp>
  void __calc_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp *__off)
- template<typename _Tp>
  bool **__cas_omp** (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _Tp>
  bool __compare_and_swap (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **__copy_tail** (std::pair< _IIter, _IIter > __b, std::pair< _IIter, _IIter > __e, _OutputIterator __r)
- void __decode2 (_CASable __x, int &__a, int &__b)
- template<typename _RAIter, typename _DifferenceTp>
  void __determine_samples (_PMWMSSortingData< _RAIter > *__sd, _DifferenceTp __num_samples)
- _CASable __encode2 (int __a, int __b)
- template<typename _DifferenceType, typename _OutputIterator>
  _OutputIterator __equally_split (_DifferenceType __n, _ThreadIndex __num_threads, _OutputIterator __s)
- template<typename _DifferenceType>
  _DifferenceType __equally_split_point (_DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __thread_no)
- template<typename _Tp>
  _Tp __fetch_and_add (volatile _Tp *__ptr, _Tp __addend)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
  std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
  std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
  std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
  std::pair< _RAIter1, _RAIter2 > __find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag)

- template<typename _IIter, typename _UserOp, typename _Functionality, typename _Red, typename _Result>
  _UserOp __for_each_template_random_access (_IIter __begin, _IIter __end, _UserOp __user_op, _Functionality &__functionality, _Red __reduction, _Result __reduction_start, _Result &__output, typename std::iterator_traits< _IIter >::difference_type __bound, _Parallelism __parallelism_tag)
- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __for_each_template_random_access_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)
- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __for_each_template_random_access_omp_loop (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)
- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __for_each_template_random_access_omp_loop_static (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)
- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __for_each_template_random_access_workstealing (_RAIter __begin, _RAIter __end, _Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)
- _ThreadIndex **__get_max_threads** ()
- bool **__is_parallel** (const _Parallelism __p)
- template<typename _IIter, typename _Compare>
  bool __is_sorted (_IIter __begin, _IIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  _RAIter __median_of_three_iterators (_RAIter __a, _RAIter __b, _RAIter __c, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>
  _OutputIterator __merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>
  _OutputIterator __merge_advance_movc (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>
  _OutputIterator __merge_advance_usual (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter3, typename _Compare>
  _RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__begin2, _RAIter1 _end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare>
  _RAIter3 __parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 _end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __max_length, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void __parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void __parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator __parallel_partial_sum (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator __parallel_partial_sum_basecase (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits< _IIter >::value_type __value)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator __parallel_partial_sum_linear (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits< _IIter >::difference_type __n)

- template<typename _RAIter, typename _Predicate>
  std::iterator_traits< _RAIter >::difference_type __parallel_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __parallel_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng=_RandomNumber())
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __parallel_random_shuffle_drs (_RAIter __begin, _RAIter __end, typename std::iterator_traits< _RAIter >::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __parallel_random_shuffle_drs_pu (_DRSSorterPU< _RAIter, _RandomNumberGenerator > *__pus)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__parallel_set_difference** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__parallel_set_intersection** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter, typename _OutputIterator, typename _Operation>
  _OutputIterator **__parallel_set_operation** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Operation __op)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__parallel_set_symmetric_difference** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__parallel_set_union** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism>
  void **__parallel_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __parallelism)
- template<typename _RAIter, typename _Compare>
  void __parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
  void __parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
  std::iterator_traits< _RAIter >::difference_type __parallel_sort_qs_divide (_RAIter __begin, _RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, typename std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)

- template<typename _RAIter, typename _Compare>
  void __parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _IIter, class _OutputIterator>
  _OutputIterator __parallel_unique_copy (_IIter __first, _IIter __last, _OutputIterator __result)
- template<typename _IIter, class _OutputIterator, class _BinaryPredicate>
  _OutputIterator __parallel_unique_copy (_IIter __first, _IIter __last, _OutputIterator __result, _BinaryPredicate __binary_pred)
- template<typename _RAIter, typename _Compare>
  void __qsb_conquer (_QSBThreadLocal< _RAIter > **__tls, _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __iam, _ThreadIndex __num_threads, bool __parent_wait)
- template<typename _RAIter, typename _Compare>
  std::iterator_traits< _RAIter >::difference_type __qsb_divide (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
  void __qsb_local_sort_with_helping (_QSBThreadLocal< _RAIter > **__tls, _Compare &__comp, _ThreadIndex __iam, bool __wait)
- template<typename _RandomNumberGenerator>
  int __random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)
- template<typename _Size>
  _Size __rd_log2 (_Size __n)
- template<typename _Tp>
  _Tp __round_up_to_pow2 (_Tp __x)
- template<typename __RAIter1, typename __RAIter2, typename _Pred>
  __RAIter1 __search_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, __RAIter2 __end2, _Pred __pred)
- template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 __sequential_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __sequential_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rng)
- template<typename _IIter>
  void __shrink (std::vector< _IIter > &__os_starts, size_t &__count_to_two, size_t &__range_length)
- template<typename _IIter>
  void __shrink_and_double (std::vector< _IIter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)
- void __yield ()
- template<typename _IIter, typename _FunctorType>
  size_t list_partition (const _IIter __begin, const _IIter __end, _IIter *__starts, size_t *__lengths, const int __num_parts, _FunctorType &__f, int __oversampling=0)
- template<typename _Tp>
  const _Tp & max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp>
  const _Tp & min (const _Tp &__a, const _Tp &__b)
- template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>
  void multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >())
- template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>
  _Tp multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp >())

- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut ↩
  __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut ↩
  __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sampling_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut ↩
  __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut ↩
  __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut ↩
  __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _↩
  DifferenceTp, typename _Compare>
  _RAIter3 multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __↩
  target, _DifferenceTp __length, _Compare __comp)
- template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _↩
  DifferenceTp, typename _Compare>
  _RAIter3 multiway_merge_4_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __↩
  target, _DifferenceTp __length, _Compare __comp)
- template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
  void multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _Difference↩
  Type __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _↩
  DifferenceType > > *__pieces)
- template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 _↩
  _target, _DifferenceTp __length, _Compare __comp)
- template<typename _UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _↩
  RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_↩
  type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, ↩
  _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value↩
  _type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
  void multiway_merge_sampling_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _↩
  DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _↩
  DifferenceType, _DifferenceType > > *__pieces)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↩
  _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _↩
  RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↩
  _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↩
  _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))

- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, ↩
  _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter, typename _Compare>
  _RAIter3 parallel_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __↩
  target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, _ThreadIndex __num_threads)
- template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
  void parallel_sort_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
  void parallel_sort_mwms_pu (_PMWMSSortingData< _RAIter > *__sd, _Compare &__comp)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _↩
  RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _↩
  RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _↩
  RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _↩
  RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _↩
  RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↩
  seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↩
  seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↩
  seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↩
  seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __↩
  seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)

Variables

- static const int _CASable_bits
- static const _CASable _CASable_mask

4.6.1 Detailed Description

GNU parallel code for public use.

4.6.2 Typedef Documentation

_BinIndex

```
typedef unsigned short __gnu_parallel::_BinIndex
```

Type to hold the index of a bin.
Since many variables of this type are allocated, it should be chosen as small as possible.

_CASable

```
typedef int64_t __gnu_parallel::_CASable
```

Longest compare-and-swappable integer type on this platform.

_SequenceIndex

```
typedef uint64_t __gnu_parallel::_SequenceIndex
```

Unsigned integer to index __elements. The total number of elements for each algorithm must fit into this type.

_ThreadIndex

```
typedef uint16_t __gnu_parallel::_ThreadIndex
```

Unsigned integer to index a thread number. The maximum thread number (for each processor) must fit into this type.

4.6.3 Enumeration Type Documentation

_AlgorithmStrategy

```
enum __gnu_parallel::_AlgorithmStrategy
```

Strategies for run-time algorithm selection:

_FindAlgorithm

```
enum __gnu_parallel::_FindAlgorithm
```

Find algorithms:

_MultiwayMergeAlgorithm

```
enum __gnu_parallel::_MultiwayMergeAlgorithm
```

Merging algorithms:

_Parallelism

```
enum __gnu_parallel::_Parallelism
```

Run-time equivalents for the compile-time tags.

Enumerator

sequential	Not parallel.
parallel_unbalanced	Parallel unbalanced (equal-sized chunks).
parallel_balanced	Parallel balanced (work-stealing).
parallel_omp_loop	Parallel with OpenMP dynamic load-balancing.
parallel_omp_loop_static	Parallel with OpenMP static load-balancing.
parallel_taskqueue	Parallel with OpenMP taskqueue construct.

_PartialSumAlgorithm

enum __gnu_parallel::_PartialSumAlgorithm
Partial sum algorithms: recursive, linear.

_SortAlgorithm

enum __gnu_parallel::_SortAlgorithm
Sorting algorithms:

_SplittingAlgorithm

enum __gnu_parallel::_SplittingAlgorithm
Sorting/merging algorithms: sampling, __exact.

4.6.4 Function Documentation

__calc_borders()

```
template<typename _RAIter, typename _DifferenceTp>
void __gnu_parallel::__calc_borders (
             _RAIter __elements,
             _DifferenceTp __length,
             _DifferenceTp * __off)
```

Precalculate __advances for Knuth-Morris-Pratt algorithm.

Parameters

__elements	Begin iterator of sequence to search for.
__length	Length of sequence to search for.
__off	Returned __offsets.

Referenced by __search_template().

__compare_and_swap()

```
template<typename _Tp>
bool __gnu_parallel::__compare_and_swap (
             volatile _Tp * __ptr,
             _Tp __comparand,
             _Tp __replacement)  [inline]
```

Compare-and-swap.
Compare `*__ptr` and `__comparand`. If equal, let `*__ptr=__replacement` and return `true`, return `false` otherwise.

Parameters

__ptr	Pointer to signed integer.
__comparand	Compare value.
__replacement	Replacement value.

Referenced by __parallel_partition(), __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_back(), and __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_front().

__decode2()

```
void __gnu_parallel::__decode2 (
            _CASable __x,
            int & __a,
            int & __b)  [inline]
```

Decode two integers from one gnu_parallel::_CASable.

Parameters

_↩ _x	__gnu_parallel::_CASable to decode integers from.
_↩ _a	First integer, to be decoded from the most-significant _CASable_bits/2 bits of __x.
_↩ _b	Second integer, to be encoded in the least-significant _CASable_bits/2 bits of __x.

See also

__encode2

References _CASable_bits, and _CASable_mask.
Referenced by __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_back(), __gnu_parallel::_RestrictedBoundedConcurr
and __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front().

__determine_samples()

```
template<typename _RAIter, typename _DifferenceTp>
void __gnu_parallel::__determine_samples (
            _PMWMSSortingData< _RAIter > * __sd,
            _DifferenceTp __num_samples)
```

Select _M_samples from a sequence.

Parameters

__sd	Pointer to algorithm data. _Result will be placed in __sd->_M_samples.
__num_samples	Number of _M_samples to select.

References __equally_split(), __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_samples, __gnu_parallel::_PMWMSSortingData<
and __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_starts.

__encode2()

```
_CASable __gnu_parallel::__encode2 (
            int __a,
            int __b)  [inline]
```

Encode two integers into one gnu_parallel::_CASable.

Parameters

_↩ _a	First integer, to be encoded in the most-significant _CASable_bits/2 bits.

_↩ _b	Second integer, to be encoded in the least-significant _CASable_bits/2 bits.

Returns

value encoding __a and __b.

See also

__decode2

References _CASable_bits.
Referenced by __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::_RestrictedBoundedConcurrentQueue(), __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_back(), __gnu_parallel::_RestrictedBoundedConcurrentQueue< _T and __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front().

__equally_split()

```
template<typename _DifferenceType, typename _OutputIterator>
_OutputIterator __gnu_parallel::__equally_split (
            _DifferenceType __n,
            _ThreadIndex __num_threads,
            _OutputIterator __s)
```

function to split a sequence into parts of almost equal size.
The resulting sequence __s of length __num_threads+1 contains the splitting positions when splitting the range [0,__n) into parts of almost equal size (plus minus 1). The first entry is 0, the last one n. There may result empty parts.

Parameters

__n	Number of elements
__num_threads	Number of parts
__s	Splitters

Returns

End of __splitter sequence, i.e. __s+__num_threads+1

Referenced by __determine_samples(), __find_template(), __parallel_partial_sum_linear(), __parallel_unique_copy(), __search_template(), and multiway_merge_exact_splitting().

__equally_split_point()

```
template<typename _DifferenceType>
_DifferenceType __gnu_parallel::__equally_split_point (
            _DifferenceType __n,
            _ThreadIndex __num_threads,
            _ThreadIndex __thread_no)
```

function to split a sequence into parts of almost equal size.
Returns the position of the splitting point between thread number __thread_no (included) and thread number __thread↩ _no+1 (excluded).

Parameters

__n	Number of elements
__num_threads	Number of parts
__thread_no	Number of threads

Returns

splitting point

Referenced by __for_each_template_random_access_ed().

__fetch_and_add()

```
template<typename _Tp>
_Tp __gnu_parallel::__fetch_and_add (
             volatile _Tp * __ptr,
             _Tp __addend)  [inline]
```

Add a value to a variable, atomically.

Parameters

__ptr	Pointer to a signed integer.
__addend	Value to add.

Referenced by __find_template(), __for_each_template_random_access_workstealing(), __parallel_partition(), and __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front().

__find_template() [1/4]

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (
             _RAIter1 __begin1,
             _RAIter1 __end1,
             _RAIter2 __begin2,
             _Pred __pred,
             _Selector __selector)  [inline]
```

Parallel std::find, switch for different algorithms.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence. Must have same length as first sequence.
__pred	Find predicate.
__selector	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

References __find_template(), __gnu_parallel::_Settings::get(), and std::make_pair().
Referenced by __find_template().

__find_template() [2/4]

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (
            _RAIter1 __begin1,
            _RAIter1 __end1,
            _RAIter2 __begin2,
            _Pred __pred,
            _Selector __selector,
            constant_size_blocks_tag )
```

Parallel std::find, constant block size variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
__pred	Find predicate.
__selector	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

See also

__gnu_parallel::_Settings::find_sequential_search_size

__gnu_parallel::_Settings::find_block_size There are two main differences between the growing blocks and the constant-size blocks variants.

1. For GB, the block size grows; for CSB, the block size is fixed.
2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

References _GLIBCXX_CALL, __gnu_parallel::_Settings::find_initial_block_size, __gnu_parallel::_Settings::find_sequential_search_size, std::pair< _T1, _T2 >::first, __gnu_parallel::_Settings::get(), and std::min().

__find_template() [3/4]

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (
            _RAIter1 __begin1,
            _RAIter1 __end1,
            _RAIter2 __begin2,
            _Pred __pred,
            _Selector __selector,
            equal_split_tag )
```

Parallel std::find, equal splitting variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
__pred	Find predicate.
__selector	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

References __equally_split(), and _GLIBCXX_CALL.

__find_template() [4/4]

```
template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (
             _RAIter1 __begin1,
             _RAIter1 __end1,
             _RAIter2 __begin2,
             _Pred __pred,
             _Selector __selector,
             growing_blocks_tag )
```

Parallel std::find, growing block size variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence. Second __sequence must have same length as first sequence.
__pred	Find predicate.
__selector	_Functionality (e. g. std::find_if(), std::equal(),...)

Returns

Place of finding in both sequences.

See also

__gnu_parallel::_Settings::find_sequential_search_size

__gnu_parallel::_Settings::find_scale_factor

There are two main differences between the growing blocks and the constant-size blocks variants.

1. For GB, the block size grows; for CSB, the block size is fixed.
2. For GB, the blocks are allocated dynamically; for CSB, the blocks are allocated in a predetermined manner, namely spacial round-robin.

References __fetch_and_add(), _GLIBCXX_CALL, __gnu_parallel::_Settings::find_scale_factor, __gnu_parallel::_Settings::find_sequentia
std::pair< _T1, _T2 >::first, __gnu_parallel::_Settings::get(), std::max(), and std::min().

__for_each_template_random_access()

```
template<typename _IIter, typename _UserOp, typename _Functionality, typename _Red, typename _Result>
_UserOp __gnu_parallel::__for_each_template_random_access (
            _IIter __begin,
            _IIter __end,
            _UserOp __user_op,
            _Functionality & __functionality,
            _Red __reduction,
            _Result __reduction_start,
            _Result & __output,
            typename std::iterator_traits< _IIter >::difference_type __bound,
            _Parallelism __parallelism_tag)
```

Chose the desired algorithm by evaluating __parallelism_tag.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__user_op	A user-specified functor (comparator, predicate, associative operator,...)
__functionality	functor to *process* an element with __user_op (depends on desired functionality, e. g. accumulate, for_each,...
__reduction	Reduction functor.
__reduction_start	Initial value for reduction.
__output	Output iterator.
__bound	Maximum number of elements processed.
__parallelism_tag	Parallelization method

References __for_each_template_random_access_ed(), __for_each_template_random_access_omp_loop(), __for_each_template_rando parallel_omp_loop, parallel_omp_loop_static, and parallel_unbalanced.

__for_each_template_random_access_ed()

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_ed (
            _RAIter __begin,
            _RAIter __end,
            _Op __o,
            _Fu & __f,
            _Red __r,
            _Result __base,
            _Result & __output,
            typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using hand-crafted parallelization by equal splitting the work.

Parameters

__begin	Begin iterator of element sequence.
__end	End iterator of element sequence.

__o	User-supplied functor (comparator, predicate, adding functor, ...)
__f	Functor to "process" an element with __op (depends on desired functionality, e. g. for std::for_each(), ...).
__r	Functor to "add" a single __result to the already processed elements (depends on functionality).
__base	Base value for reduction.
__output	Pointer to position where final result is written to
__bound	Maximum number of elements processed (e. g. for std::count_n()).

Returns

User-supplied functor (that may contain a part of the result).

References __equally_split_point(), and min().
Referenced by __for_each_template_random_access().

__for_each_template_random_access_omp_loop()

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_omp_loop (
            _RAIter __begin,
            _RAIter __end,
            _Op __o,
            _Fu & __f,
            _Red __r,
            _Result __base,
            _Result & __output,
            typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop.

Parameters

__begin	Begin iterator of element sequence.
__end	End iterator of element sequence.
__o	User-supplied functor (comparator, predicate, adding functor, etc.).
__f	Functor to *process* an element with __op (depends on desired functionality, e. g. for std::for_each(), ...).

__r	Functor to *add* a single __result to the already processed elements (depends on functionality).
__base	Base value for reduction.
__output	Pointer to position where final result is written to
__bound	Maximum number of elements processed (e. g. for std::count_n()).

Returns

User-supplied functor (that may contain a part of the result).

References min().
Referenced by __for_each_template_random_access().

__for_each_template_random_access_omp_loop_static()

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (
              _RAIter __begin,
              _RAIter __end,
              _Op __o,
              _Fu & __f,
              _Red __r,
              _Result __base,
              _Result & __output,
              typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Embarrassingly parallel algorithm for random access iterators, using an OpenMP for loop with static scheduling.

Parameters

__begin	Begin iterator of element sequence.
__end	End iterator of element sequence.
__o	User-supplied functor (comparator, predicate, adding functor, ...).
__f	Functor to *process* an element with __op (depends on desired functionality, e. g. for std::for_each(), ...).
__r	Functor to *add* a single __result to the already processed __elements (depends on functionality).
__base	Base value for reduction.
__output	Pointer to position where final result is written to
__bound	Maximum number of elements processed (e. g. for std::count_n()).

Returns

User-supplied functor (that may contain a part of the result).

References std::min().

__for_each_template_random_access_workstealing()

```
template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
_Op __gnu_parallel::__for_each_template_random_access_workstealing (
            _RAIter __begin,
            _RAIter __end,
            _Op __op,
            _Fu & __f,
            _Red __r,
            _Result __base,
            _Result & __output,
            typename std::iterator_traits< _RAIter >::difference_type __bound)
```

Work stealing algorithm for random access iterators.
Uses O(1) additional memory. Synchronization at job lists is done with atomic operations.

Parameters

__begin	Begin iterator of element sequence.
__end	End iterator of element sequence.
__op	User-supplied functor (comparator, predicate, adding functor, ...).
__f	Functor to *process* an element with __op (depends on desired functionality, e. g. for std::for_each(), ...).
__r	Functor to *add* a single __result to the already processed elements (depends on functionality).
__base	Base value for reduction.
__output	Pointer to position where final result is written to
__bound	Maximum number of elements processed (e. g. for std::count_n()).

Returns

User-supplied functor (that may contain a part of the result).

References __fetch_and_add(), __yield(), _GLIBCXX_CALL, __gnu_parallel::_Job< _DifferenceTp >::_M_first, __gnu_parallel::_Job< _DifferenceTp >::_M_last, __gnu_parallel::_Job< _DifferenceTp >::_M_load, __gnu_parallel::_Settings::cache_l __gnu_parallel::_Settings::get(), max(), and min().
Referenced by __for_each_template_random_access().

__is_sorted()

```
template<typename _IIter, typename _Compare>
bool __gnu_parallel::__is_sorted (
```

```
            _IIter __begin,
            _IIter __end,
            _Compare __comp)
```

Check whether [__begin, __end) is sorted according to __comp.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__comp	Comparator.

Returns

true if sorted, false otherwise.

Referenced by __sequential_multiway_merge(), multiway_merge_loser_tree_sentinel(), and parallel_multiway_merge().

__median_of_three_iterators()

```
template<typename _RAIter, typename _Compare>
_RAIter __gnu_parallel::__median_of_three_iterators (
            _RAIter __a,
            _RAIter __b,
            _RAIter __c,
            _Compare __comp)
```

Compute the median of three referenced elements, according to __comp.

Parameters

__a	First iterator.
__b	Second iterator.
__c	Third iterator.
__comp	Comparator.

Referenced by __qsb_divide().

__merge_advance()

```
template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp,
typename _Compare>
_OutputIterator __gnu_parallel::__merge_advance (
            _RAIter1 & __begin1,
            _RAIter1 __end1,
            _RAIter2 & __begin2,
            _RAIter2 __end2,
            _OutputIterator __target,
            _DifferenceTp __max_length,
            _Compare __comp)  [inline]
```

Merge routine being able to merge only the __max_length smallest elements.

The __begin iterators are advanced accordingly, they might not reach __end, in contrast to the usual variant. Static switch on whether to use the conditional-move variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

References __merge_advance_movc(), and _GLIBCXX_CALL.
Referenced by __parallel_merge_advance(), and __sequential_multiway_merge().

__merge_advance_movc()

```
template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp,
typename _Compare>
_OutputIterator __gnu_parallel::__merge_advance_movc (
             _RAIter1 & __begin1,
             _RAIter1 __end1,
             _RAIter2 & __begin2,
             _RAIter2 __end2,
             _OutputIterator __target,
             _DifferenceTp __max_length,
             _Compare __comp)
```

Merge routine being able to merge only the __max_length smallest elements.
The __begin iterators are advanced accordingly, they might not reach __end, in contrast to the usual variant. Specially designed code should allow the compiler to generate conditional moves instead of branches.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

Referenced by __merge_advance().

__merge_advance_usual()

```
template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp,
typename _Compare>
_OutputIterator __gnu_parallel::__merge_advance_usual (
             _RAIter1 & __begin1,
             _RAIter1 __end1,
             _RAIter2 & __begin2,
             _RAIter2 __end2,
             _OutputIterator __target,
             _DifferenceTp __max_length,
             _Compare __comp)
```

Merge routine being able to merge only the __max_length smallest elements.
The __begin iterators are advanced accordingly, they might not reach __end, in contrast to the usual variant.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

__parallel_merge_advance() [1/2]

```
template<typename _RAIter1, typename _RAIter3, typename _Compare>
_RAIter3 __gnu_parallel::__parallel_merge_advance (
             _RAIter1 & __begin1,
             _RAIter1 __end1,
             _RAIter1 & __begin2,
             _RAIter1 __end2,
             _RAIter3 __target,
             typename std::iterator_traits< _RAIter1 >::difference_type __max_length,
             _Compare __comp)  [inline]
```

Parallel merge routine being able to merge only the __max_length smallest elements.
The __begin iterators are advanced accordingly, they might not reach __end, in contrast to the usual variant. The functionality is projected onto parallel_multiway_merge.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

References std::make_pair(), multiway_merge_exact_splitting(), and parallel_multiway_merge().

__parallel_merge_advance() [2/2]

```
template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare>
_RAIter3 __gnu_parallel::__parallel_merge_advance (
             _RAIter1 & __begin1,
             _RAIter1 __end1,
             _RAIter2 & __begin2,
             _RAIter2 __end2,
             _RAIter3 __target,
             typename std::iterator_traits< _RAIter1 >::difference_type __max_length,
             _Compare __comp)  [inline]
```

Merge routine fallback to sequential in case the iterators of the two input sequences are of different type.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__target	Target begin iterator.
__max_length	Maximum number of elements to merge.
__comp	Comparator.

Returns

Output end iterator.

References __merge_advance().

__parallel_nth_element()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_nth_element (
             _RAIter __begin,
```

```
            _RAIter __nth,
            _RAIter __end,
            _Compare __comp)
```

Parallel implementation of std::nth_element().

Parameters

__begin	Begin iterator of input sequence.
__nth	_Iterator of element that must be in position afterwards.
__end	End iterator of input sequence.
__comp	Comparator.

References __parallel_partition(), _GLIBCXX_CALL, __gnu_parallel::_Settings::get(), std::max(), __gnu_parallel::_Settings::nth_element_
and __gnu_parallel::_Settings::partition_minimal_n.
Referenced by __parallel_partial_sort().

__parallel_partial_sort()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_partial_sort (
            _RAIter __begin,
            _RAIter __middle,
            _RAIter __end,
            _Compare __comp)
```

Parallel implementation of std::partial_sort().

Parameters

__begin	Begin iterator of input sequence.
__middle	Sort until this position.
__end	End iterator of input sequence.
__comp	Comparator.

References __parallel_nth_element().

__parallel_partial_sum()

```
template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator __gnu_parallel::__parallel_partial_sum (
            _IIter __begin,
            _IIter __end,
            _OutputIterator __result,
            _BinaryOperation __bin_op)
```

Parallel partial sum front-__end.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__result	Begin iterator of output sequence.

__bin_op	Associative binary function.

Returns

End iterator of output sequence.

References __parallel_partial_sum_linear(), _GLIBCXX_CALL, and __gnu_parallel::_Settings::get().

__parallel_partial_sum_basecase()

```
template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (
            _IIter __begin,
            _IIter __end,
            _OutputIterator __result,
            _BinaryOperation __bin_op,
            typename std::iterator_traits< _IIter >::value_type __value)
```

Base case prefix sum routine.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__result	Begin iterator of output sequence.
__bin_op	Associative binary function.
__value	Start value. Must be passed since the neutral element is unknown in general.

Returns

End iterator of output sequence.

Referenced by __parallel_partial_sum_linear().

__parallel_partial_sum_linear()

```
template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
_OutputIterator __gnu_parallel::__parallel_partial_sum_linear (
            _IIter __begin,
            _IIter __end,
            _OutputIterator __result,
            _BinaryOperation __bin_op,
            typename std::iterator_traits< _IIter >::difference_type __n)
```

Parallel partial sum implementation, two-phase approach, no recursion.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__result	Begin iterator of output sequence.
__bin_op	Associative binary function.

__n	Length of sequence.

Returns

End iterator of output sequence.

References __equally_split(), __parallel_partial_sum_basecase(), __gnu_parallel::_Settings::get(), std::max(), std::min(), and __gnu_parallel::_Settings::partial_sum_dilation.
Referenced by __parallel_partial_sum().

__parallel_partition()

```
template<typename _RAIter, typename _Predicate>
std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_partition (
            _RAIter __begin,
            _RAIter __end,
            _Predicate __pred,
            _ThreadIndex __num_threads)
```

Parallel implementation of std::partition.

Parameters

__begin	Begin iterator of input sequence to split.
__end	End iterator of input sequence to split.
__pred	Partition predicate, possibly including some kind of pivot.
__num_threads	Maximum number of threads to use for this task.

Returns

Number of elements not fulfilling the predicate.

References __compare_and_swap(), __fetch_and_add(), _GLIBCXX_CALL, _GLIBCXX_VOLATILE, __gnu_parallel::_Settings::get(), std::max(), __gnu_parallel::_Settings::partition_chunk_share, and __gnu_parallel::_Settings::partition_chunk_size.
Referenced by __parallel_nth_element(), __parallel_sort_qs_divide(), and __qsb_divide().

__parallel_random_shuffle()

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__parallel_random_shuffle (
            _RAIter __begin,
            _RAIter __end,
            _RandomNumberGenerator __rng = _RandomNumber())  [inline]
```

Parallel random public call.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__rng	Random number generator to use.

References __parallel_random_shuffle_drs().

__parallel_random_shuffle_drs()

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__parallel_random_shuffle_drs (
            _RAIter __begin,
            _RAIter __end,
            typename std::iterator_traits< _RAIter >::difference_type __n,
            _ThreadIndex __num_threads,
            _RandomNumberGenerator & __rng)
```

Main parallel random shuffle step.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__n	Length of sequence.
__num_threads	Number of threads to use.
__rng	Random number generator to use.

References __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__bins_end, __parallel_random_shuffle_drs_pu(), __rd_log2(), __round_up_to_pow2(), __sequential_random_shuffle(), _GLIBCXX_CALL, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_bins_begin, __gnu_parallel::_DRandomShufflingGlobalDa __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_bins_begin, __gnu_parallel::_DRandomShufflingGlobalDa __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter > __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads, __gnu_parallel::_DRSSorterPU< _RAIter, __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed, __gnu_parallel::_DRandomShufflingGlobalData< _ __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_temporaries, __gnu_parallel::_Settings::get(), __gnu_parallel::_Settings::L2_cache_size, std::max(), std::numeric_limits< _Tp >::max(), std::min(), and __gnu_parallel::_Settings::TLB_

Referenced by __parallel_random_shuffle().

__parallel_random_shuffle_drs_pu()

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__parallel_random_shuffle_drs_pu (
            _DRSSorterPU< _RAIter, _RandomNumberGenerator > * __pus)
```

Random shuffle code executed by each thread.

Parameters

__pus	Array of thread-local data records.

References __random_number_pow2(), __sequential_random_shuffle(), __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_bins_begin, __gnu_parallel::_DRandomShufflingGlobalDa __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter > __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads, __gnu_parallel::_DRSSorterPU< _RAIter, __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed, __gnu_parallel::_DRandomShufflingGlobalData< _ __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts, __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M and std::partial_sum().

Referenced by __parallel_random_shuffle_drs().

__parallel_sort() [1/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
```

```
void __gnu_parallel::__parallel_sort (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            balanced_quicksort_tag __parallelism)  [inline]
```

Choose balanced quicksort for parallel sorting.

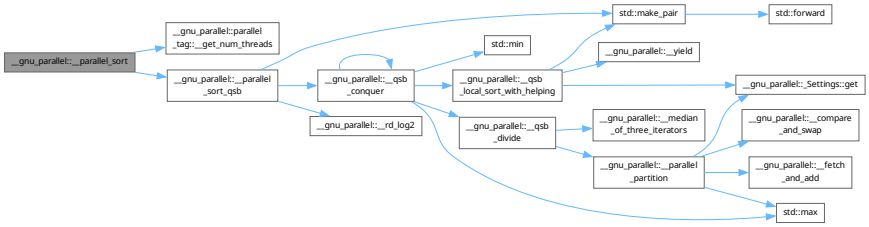
Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), __parallel_sort_qsb(), and _GLIBCXX_CALL.
Here is the call graph for this function:



__parallel_sort() [2/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            default_parallel_tag __parallelism)  [inline]
```

Choose multiway mergesort with exact splitting, for parallel sorting.

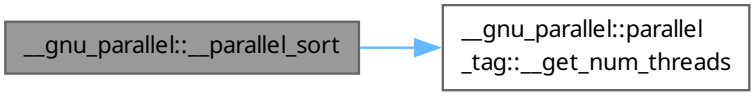
Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), and _GLIBCXX_CALL.
Here is the call graph for this function:



__parallel_sort() [3/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            multiway_mergesort_exact_tag __parallelism)  [inline]
```

Choose multiway mergesort with exact splitting, for parallel sorting.

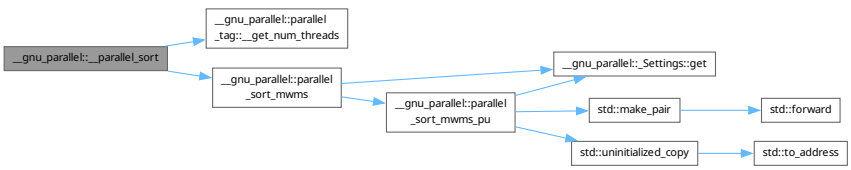
Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), _GLIBCXX_CALL, and parallel_sort_mwms().
Here is the call graph for this function:



__parallel_sort() [4/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
             _RAIter __begin,
             _RAIter __end,
             _Compare __comp,
             multiway_mergesort_sampling_tag __parallelism)  [inline]
```

Choose multiway mergesort with splitting by sampling, for parallel sorting.

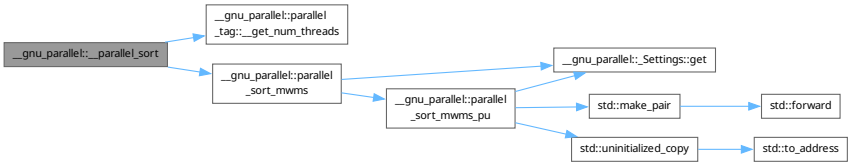
Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), _GLIBCXX_CALL, and parallel_sort_mwms().
Here is the call graph for this function:



__parallel_sort() [5/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
             _RAIter __begin,
             _RAIter __end,
             _Compare __comp,
             multiway_mergesort_tag __parallelism)  [inline]
```

Choose multiway mergesort, splitting variant at run-time, for parallel sorting.

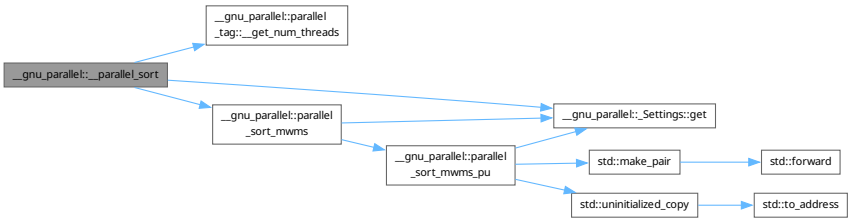
Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), _GLIBCXX_CALL, __gnu_parallel::_Settings::get(), and parallel_sort_mwms().
Here is the call graph for this function:



__parallel_sort() [6/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            parallel_tag __parallelism)  [inline]
```

Choose a parallel sorting algorithm.

Parameters

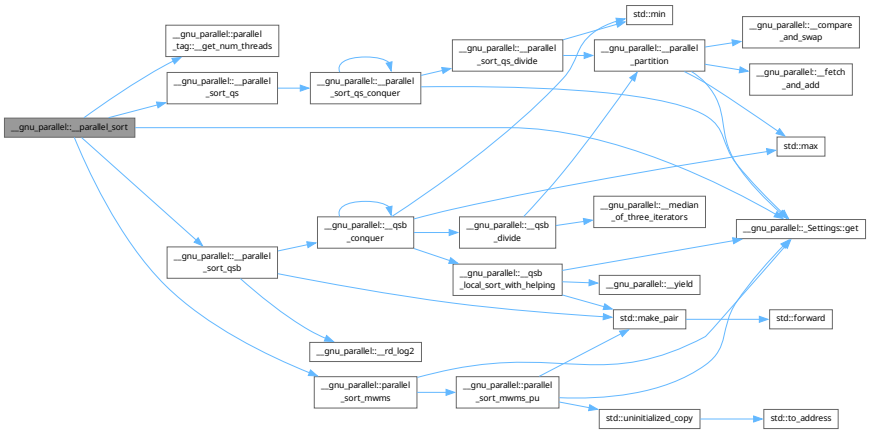
__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), __parallel_sort_qs(), __parallel_sort_qsb(), _GLIBCXX_CALL, __gnu_parallel::_Settings::get(), and parallel_sort_mwms().

Here is the call graph for this function:



__parallel_sort() [7/7]

```
template<bool __stable, typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            quicksort_tag __parallelism)  [inline]
```

Choose quicksort for parallel sorting.

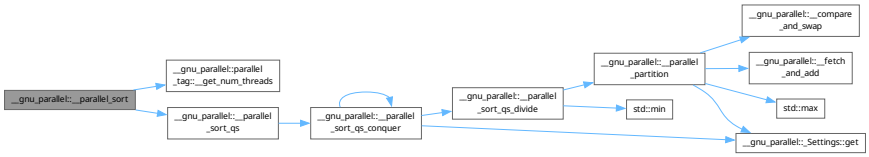
Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__comp	Comparator.

Template Parameters

__stable	Sort stable.

References __gnu_parallel::parallel_tag::__get_num_threads(), __parallel_sort_qs(), and _GLIBCXX_CALL.
Here is the call graph for this function:



__parallel_sort_qs()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort_qs (
             _RAIter __begin,
             _RAIter __end,
             _Compare __comp,
             _ThreadIndex __num_threads)
```

Unbalanced quicksort main call.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator input sequence, ignored.
__comp	Comparator.
__num_threads	Number of threads that are allowed to work on this part.

References __parallel_sort_qs_conquer(), and _GLIBCXX_CALL.
Referenced by __parallel_sort(), and __parallel_sort().

__parallel_sort_qs_conquer()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort_qs_conquer (
             _RAIter __begin,
             _RAIter __end,
             _Compare __comp,
             _ThreadIndex __num_threads)
```

Unbalanced quicksort conquer step.

Parameters

__begin	Begin iterator of subsequence.
__end	End iterator of subsequence.
__comp	Comparator.
__num_threads	Number of threads that are allowed to work on this part.

References __parallel_sort_qs_conquer(), __parallel_sort_qs_divide(), and __gnu_parallel::_Settings::get().
Referenced by __parallel_sort_qs(), and __parallel_sort_qs_conquer().

__parallel_sort_qs_divide()

```
template<typename _RAIter, typename _Compare>
std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_sort_qs_divide (
             _RAIter __begin,
             _RAIter __end,
             _Compare __comp,
             typename std::iterator_traits< _RAIter >::difference_type __pivot_rank,
             typename std::iterator_traits< _RAIter >::difference_type __num_samples,
             _ThreadIndex __num_threads)
```

Unbalanced quicksort divide step.

Parameters

__begin	Begin iterator of subsequence.
__end	End iterator of subsequence.
__comp	Comparator.
__pivot_rank	Desired __rank of the pivot.
__num_samples	Choose pivot from that many samples.
__num_threads	Number of threads that are allowed to work on this part.

References __parallel_partition(), and std::min().
Referenced by __parallel_sort_qs_conquer().

__parallel_sort_qsb()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__parallel_sort_qsb (
             _RAIter __begin,
             _RAIter __end,
             _Compare __comp,
             _ThreadIndex __num_threads)
```

Top-level quicksort routine.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__comp	Comparator.
__num_threads	Number of threads that are allowed to work on this part.

References __qsb_conquer(), __rd_log2(), _GLIBCXX_CALL, and std::make_pair().
Referenced by __parallel_sort(), and __parallel_sort().

__parallel_unique_copy() [1/2]

```
template<typename _IIter, class _OutputIterator>
_OutputIterator __gnu_parallel::__parallel_unique_copy (
             _IIter __first,
             _IIter __last,
             _OutputIterator __result)  [inline]
```

Parallel std::unique_copy(), without explicit equality predicate.

Parameters

__first	Begin iterator of input sequence.
__last	End iterator of input sequence.
__result	Begin iterator of result __sequence.

Returns

End iterator of result __sequence.

References __parallel_unique_copy().

__parallel_unique_copy() [2/2]

```
template<typename _IIter, class _OutputIterator, class _BinaryPredicate>
_OutputIterator __gnu_parallel::__parallel_unique_copy (
              _IIter __first,
              _IIter __last,
              _OutputIterator __result,
              _BinaryPredicate __binary_pred)
```

Parallel std::unique_copy(), w/__o explicit equality predicate.

Parameters

__first	Begin iterator of input sequence.
__last	End iterator of input sequence.
__result	Begin iterator of result __sequence.
__binary_pred	Equality predicate.

Returns

End iterator of result __sequence.

References __equally_split(), and _GLIBCXX_CALL.
Referenced by __parallel_unique_copy().

__qsb_conquer()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__qsb_conquer (
              _QSBThreadLocal< _RAIter > ** __tls,
              _RAIter __begin,
              _RAIter __end,
              _Compare __comp,
              _ThreadIndex __iam,
              _ThreadIndex __num_threads,
              bool __parent_wait)
```

Quicksort conquer step.

Parameters

__tls	Array of thread-local storages.
__begin	Begin iterator of subsequence.
__end	End iterator of subsequence.
__comp	Comparator.
__iam	Number of the thread processing this function.
__num_threads	Number of threads that are allowed to work on this part.

References __qsb_conquer(), __qsb_divide(), __qsb_local_sort_with_helping(), __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_eler
__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial, std::max(), and std::min().
Referenced by __parallel_sort_qsb(), and __qsb_conquer().

__qsb_divide()

```
template<typename _RAIter, typename _Compare>
std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__qsb_divide (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            _ThreadIndex __num_threads)
```

Balanced quicksort divide step.

Parameters

__begin	Begin iterator of subsequence.
__end	End iterator of subsequence.
__comp	Comparator.
__num_threads	Number of threads that are allowed to work on this part.

Precondition

(__end-__begin)>=1

References __median_of_three_iterators(), and __parallel_partition().
Referenced by __qsb_conquer().

__qsb_local_sort_with_helping()

```
template<typename _RAIter, typename _Compare>
void __gnu_parallel::__qsb_local_sort_with_helping (
            _QSBThreadLocal< _RAIter > ** __tls,
            _Compare & __comp,
            _ThreadIndex __iam,
            bool __wait)
```

Quicksort step doing load-balanced local sort.

Parameters

__tls	Array of thread-local storages.
__comp	Comparator.
__iam	Number of the thread processing this function.

References __yield(), _GLIBCXX_PARALLEL_ASSERTIONS, __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_elements_leftover,
__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial, __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_leftover_parts,
__gnu_parallel::_QSBThreadLocal< _RAIter >::_M_num_threads, __gnu_parallel::_Settings::get(), std::make_pair(),
and __gnu_parallel::_Settings::sort_qsb_base_case_maximal_n.
Referenced by __qsb_conquer().

__random_number_pow2()

```
template<typename _RandomNumberGenerator>
```

```
int __gnu_parallel::__random_number_pow2 (
            int __logp,
            _RandomNumberGenerator & __rng)  [inline]
```

Generate a random number in [0,2^__logp).

Parameters

__logp	Logarithm (basis 2) of the upper range __bound.
__rng	Random number generator to use.

Referenced by __parallel_random_shuffle_drs_pu(), and __sequential_random_shuffle().

__rd_log2()

```
template<typename _Size>
_Size __gnu_parallel::__rd_log2 (
            _Size __n)  [inline]
```

Calculates the rounded-down logarithm of __n for base 2.

Parameters

_↩ _n	Argument.

Returns

Returns 0 for any argument <1.

Referenced by __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase(), __parallel_random_shuffle_drs(), __parallel_sort_qsb(), __round_up_to_pow2(), __sequential_random_shuffle(), multiseq_partition(), and multiseq_selection().

__round_up_to_pow2()

```
template<typename _Tp>
_Tp __gnu_parallel::__round_up_to_pow2 (
            _Tp __x)
```

Round up to the next greater power of 2.

Parameters

_↩ _x	_Integer to round up

References __rd_log2().
Referenced by __parallel_random_shuffle_drs(), __sequential_random_shuffle(), and multiseq_selection().

__search_template()

```
template<typename __RAIter1, typename __RAIter2, typename _Pred>
__RAIter1 __gnu_parallel::__search_template (
            __RAIter1 __begin1,
            __RAIter1 __end1,
            __RAIter2 __begin2,
```

```
            __RAIter2 __end2,
            _Pred __pred)
```

Parallel std::search.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__end2	End iterator of second sequence.
__pred	Find predicate.

Returns

Place of finding in first sequences.

References __calc_borders(), __equally_split(), _GLIBCXX_CALL, std::max(), and std::min().

__sequential_multiway_merge()

```
template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename ↩
_DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::__sequential_multiway_merge (
            _RAIterIterator __seqs_begin,
            _RAIterIterator __seqs_end,
            _RAIter3 __target,
            const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
            _DifferenceTp __length,
            _Compare __comp)
```

Sequential multi-way merging switch.
The _GLIBCXX_PARALLEL_DECISION is based on the branching factor and runtime settings.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, possibly larger than the number of elements available.
__sentinel	The sequences have __a __sentinel element.

Returns

End iterator of output sequence.

References __is_sorted(), __merge_advance(), _GLIBCXX_CALL, _GLIBCXX_PARALLEL_LENGTH, and std::min().
Referenced by multiway_merge(), multiway_merge_sentinels(), and parallel_multiway_merge().

__sequential_random_shuffle()

```
template<typename _RAIter, typename _RandomNumberGenerator>
void __gnu_parallel::__sequential_random_shuffle (
             _RAIter __begin,
             _RAIter __end,
             _RandomNumberGenerator & __rng)
```

Sequential cache-efficient random shuffle.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__rng	Random number generator to use.

References __random_number_pow2(), __rd_log2(), __round_up_to_pow2(), __sequential_random_shuffle(), __gnu_parallel::_Settings::get(), __gnu_parallel::_Settings::L2_cache_size, std::max(), std::min(), std::partial_sum(), and __gnu_parallel::_Settings::TLB_size.
Referenced by __parallel_random_shuffle_drs(), __parallel_random_shuffle_drs_pu(), and __sequential_random_shuffle().

__shrink()

```
template<typename _IIter>
void __gnu_parallel::__shrink (
             std::vector< _IIter > & __os_starts,
             size_t & __count_to_two,
             size_t & __range_length)
```

Combines two ranges into one and thus halves the number of ranges.

Parameters

__os_starts	Start positions worked on (oversampled).
__count_to_two	Counts up to 2.
__range_length	Current length of a chunk.

References std::vector< _Tp, _Alloc >::size().
Referenced by __shrink_and_double().

__shrink_and_double()

```
template<typename _IIter>
void __gnu_parallel::__shrink_and_double (
             std::vector< _IIter > & __os_starts,
             size_t & __count_to_two,
             size_t & __range_length,
             const bool __make_twice)
```

Shrinks and doubles the ranges.

Parameters

__os_starts	Start positions worked on (oversampled).
__count_to_two	Counts up to 2.

__range_length	Current length of a chunk.
__make_twice	Whether the __os_starts is allowed to be grown or not

References __shrink(), std::vector< _Tp, _Alloc >::resize(), and std::vector< _Tp, _Alloc >::size().
Referenced by list_partition().

__yield()

```
void __gnu_parallel::__yield ()  [inline]
```

Yield control to another thread, without waiting for the end of the time slice.
Referenced by __for_each_template_random_access_workstealing(), and __qsb_local_sort_with_helping().

list_partition()

```
template<typename _IIter, typename _FunctorType>
size_t __gnu_parallel::list_partition (
              const _IIter __begin,
              const _IIter __end,
              _IIter * __starts,
              size_t * __lengths,
              const int __num_parts,
              _FunctorType & __f,
              int __oversampling = 0)
```

Splits a sequence given by input iterators into parts of almost equal size.
The function needs only one pass over the sequence.

Parameters

__begin	Begin iterator of input sequence.
__end	End iterator of input sequence.
__starts	Start iterators for the resulting parts, dimension __num_parts+1. For convenience, __starts [__num_parts] contains the end iterator of the sequence.
__lengths	Length of the resulting parts.
__num_parts	Number of parts to split the sequence into.
__f	Functor to be applied to each element by traversing __it
__oversampling	Oversampling factor. If 0, then the partitions will differ in at most $\sqrt{\text{end} - \text{begin}}$ elements. Otherwise, the ratio between the longest and the shortest part is bounded by $1/(\text{oversampling} \cdot \text{num_parts})$

Returns

Length of the whole sequence.

References __shrink_and_double(), and std::vector< _Tp, _Alloc >::size().

max()

```
template<typename _Tp>
```

```
const _Tp & __gnu_parallel::max (
            const _Tp & __a,
            const _Tp & __b)  [inline]
```

Equivalent to std::max.
Referenced by __for_each_template_random_access_workstealing().

min()

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

Equivalent to std::min.
Referenced by __for_each_template_random_access_ed(), __for_each_template_random_access_omp_loop(), and __for_each_template_random_access_workstealing().

multiseq_partition()

```
template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>
void __gnu_parallel::multiseq_partition (
            _RanSeqs __begin_seqs,
            _RanSeqs __end_seqs,
            _RankType __rank,
            _RankIterator __begin_offsets,
            _Compare __comp = std::less< typename std::iterator_traits<typename std::iterator_traits<↩
_RanSeqs>::value_type:: first_type>::value_type>())
```

Splits several sorted sequences at a certain global __rank, resulting in a splitting point for each sequence. The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty. If there are several equal elements across the split, the ones on the __left side will be chosen from sequences with smaller number.

Parameters

__begin_seqs	Begin of the sequence of iterator pairs.
__end_seqs	End of the sequence of iterator pairs.
__rank	The global rank to partition at.
__begin_offsets	A random-access __sequence __begin where the __result will be stored in. Each element of the sequence is an iterator that points to the first element on the greater part of the respective __sequence.
__comp	The ordering functor, defaults to std::less<_Tp>.

References __rd_log2(), _GLIBCXX_CALL, std::vector< _Tp, _Alloc >::begin(), std::distance(), std::priority_queue< _Tp, _Sequence, _C
std::vector< _Tp, _Alloc >::end(), std::make_pair(), std::max(), std::min(), std::priority_queue< _Tp, _Sequence, _Compare >::pop(),
std::priority_queue< _Tp, _Sequence, _Compare >::push(), std::vector< _Tp, _Alloc >::push_back(), and std::priority_queue< _Tp, _Se
Referenced by multiway_merge_exact_splitting().

multiseq_selection()

```
template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>
_Tp __gnu_parallel::multiseq_selection (
            _RanSeqs __begin_seqs,
            _RanSeqs __end_seqs,
            _RankType __rank,
```

```
            _RankType & __offset,
            _Compare __comp = std::less<_Tp>())
```

Selects the element at a certain global __rank from several sorted sequences.
The sequences are passed via a sequence of random-access iterator pairs, none of the sequences may be empty.

Parameters

__begin_seqs	Begin of the sequence of iterator pairs.
__end_seqs	End of the sequence of iterator pairs.
__rank	The global rank to partition at.
__offset	The rank of the selected element in the global subsequence of elements equal to the selected element. If the selected element is unique, this number is 0.
__comp	The ordering functor, defaults to std::less.

References __rd_log2(), __round_up_to_pow2(), _GLIBCXX_CALL, std::vector< _Tp, _Alloc >::begin(), std::distance(), std::priority_queue< _Tp, _Sequence, _Compare >::empty(), std::vector< _Tp, _Alloc >::end(), std::make_pair(), std::max(), std::min(), std::priority_queue< _Tp, _Sequence, _Compare >::pop(), std::priority_queue< _Tp, _Sequence, _Compare >::pu
std::vector< _Tp, _Alloc >::push_back(), and std::priority_queue< _Tp, _Sequence, _Compare >::top().

multiway_merge()

```
template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _↩
Compare>
_RAIterOut __gnu_parallel::multiway_merge (
            _RAIterPairIterator __seqs_begin,
            _RAIterPairIterator __seqs_end,
            _RAIterOut __target,
            _DifferenceTp __length,
            _Compare __comp,
            __gnu_parallel::sequential_tag )
```

Multiway Merge Frontend.
Merge the sequences specified by seqs_begin and __seqs_end into __target. __seqs_begin and __seqs_end must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the _M_end of the same sequence in their second entry.
Ties are broken arbitrarily. See stable_multiway_merge for a variant that breaks ties by sequence number but is slower.
The first entries of the pairs (i.e. the begin iterators) will be moved forward.
The output sequence has to provide enough space for all elements that are written to it.
This function will merge the input sequences:

- not stable
- parallel, depending on the input size and Settings
- using sampling for splitting
- not using sentinels

Example:

```
int sequences[10][10];
for (int __i = 0; __i < 10; ++__i)
  for (int __j = 0; __i < 10; ++__j)
    sequences[__i][__j] = __j;

int __out[33];
```

```
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
  { seqs.push(std::make_pair<int*>(sequences[__i],
                                   sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

See also

stable_multiway_merge

Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

Postcondition

[__target, return __value) contains merged __elements from the input sequences.

return __value - __target = min(__length, number of elements in all sequences).

Template Parameters

_RAIterPairIterator	iterator over sequence of pairs of iterators
_RAIterOut	iterator over target sequence
_DifferenceTp	difference type for the sequence
_Compare	strict weak ordering type to compare elements in sequences

Parameters

__seqs_begin	__begin of sequence __sequence
__seqs_end	_M_end of sequence __sequence
__target	target sequence to merge to.
__comp	strict weak ordering to use for element comparison.
__length	Maximum length to merge, possibly larger than the number of elements available.

Returns

_M_end iterator of output sequence

References __sequential_multiway_merge(), and _GLIBCXX_CALL.

multiway_merge_3_variant()

```
template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename
_RAIter3, typename _DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::multiway_merge_3_variant (
            _RAIterIterator __seqs_begin,
```

```
            _RAIterIterator __seqs_end,
            _RAIter3 __target,
            _DifferenceTp __length,
            _Compare __comp)
```

Highly efficient 3-way merging procedure.
Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into labels in C++).
This works well for merging up to 4 sequences.
Note that making the merging stable does *not* come at a performance hit.
Whether the merging is done guarded or unguarded is selected by the used iterator class.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

References _GLIBCXX_CALL.

multiway_merge_4_variant()

```
template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename
_RAIter3, typename _DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::multiway_merge_4_variant (
            _RAIterIterator __seqs_begin,
            _RAIterIterator __seqs_end,
            _RAIter3 __target,
            _DifferenceTp __length,
            _Compare __comp)
```

Highly efficient 4-way merging procedure.
Merging is done with the algorithm implementation described by Peter Sanders. Basically, the idea is to minimize the number of necessary comparison after merging an element. The implementation trick that makes this fast is that the order of the sequences is stored in the instruction pointer (translated into goto labels in C++).
This works well for merging up to 4 sequences.
Note that making the merging stable does *not* come at a performance hit.
Whether the merging is done guarded or unguarded is selected by the used iterator class.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.

__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

References _GLIBCXX_CALL.

multiway_merge_exact_splitting()

```
template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
void __gnu_parallel::multiway_merge_exact_splitting (
            _RAIterIterator __seqs_begin,
            _RAIterIterator __seqs_end,
            _DifferenceType __length,
            _DifferenceType __total_length,
            _Compare __comp,
            std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)
```

Exact splitting for parallel multiway-merge routine.
None of the passed sequences may be empty.
References __equally_split(), _GLIBCXX_PARALLEL_LENGTH, std::vector< _Tp, _Alloc >::begin(), std::vector< _Tp, _Alloc >::end(), multiseq_partition(), and std::vector< _Tp, _Alloc >::resize().
Referenced by __parallel_merge_advance().

multiway_merge_loser_tree()

```
template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename
_Compare>
_RAIter3 __gnu_parallel::multiway_merge_loser_tree (
            _RAIterIterator __seqs_begin,
            _RAIterIterator __seqs_end,
            _RAIter3 __target,
            _DifferenceTp __length,
            _Compare __comp)
```

Multi-way merging procedure for a high branching factor, guarded case.
This merging variant uses a LoserTree class as selected by _LT.
Stability is selected through the used LoserTree class _LT.
At least one non-empty sequence is required.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

References _GLIBCXX_CALL, and _GLIBCXX_PARALLEL_LENGTH.

multiway_merge_loser_tree_sentinel()

```
template<typename _UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _↩
DifferenceTp, typename _Compare>
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (
            _RAIterIterator __seqs_begin,
            _RAIterIterator __seqs_end,
            _RAIter3 __target,
            const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
            _DifferenceTp __length,
            _Compare __comp)
```

Multi-way merging procedure for a high branching factor, requiring sentinels to exist.

Template Parameters

_UnguardedLoserTree	Loser Tree variant to use for the unguarded merging.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

References __is_sorted(), _GLIBCXX_CALL, and multiway_merge_loser_tree_unguarded().

multiway_merge_loser_tree_unguarded()

```
template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename
_Compare>
_RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (
            _RAIterIterator __seqs_begin,
            _RAIterIterator __seqs_end,
            _RAIter3 __target,
            const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator
>::value_type::first_type >::value_type & __sentinel,
            _DifferenceTp __length,
            _Compare __comp)
```

Multi-way merging procedure for a high branching factor, unguarded case.
Merging is done using the LoserTree class `_LT`.
Stability is selected by the used LoserTrees.

Precondition

No input will run out of elements during the merge.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, less equal than the total number of elements available.

Returns

End iterator of output sequence.

References _GLIBCXX_CALL.
Referenced by multiway_merge_loser_tree_sentinel().

multiway_merge_sampling_splitting()

```
template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
void __gnu_parallel::multiway_merge_sampling_splitting (
             _RAIterIterator __seqs_begin,
             _RAIterIterator __seqs_end,
             _DifferenceType __length,
             _DifferenceType __total_length,
             _Compare __comp,
             std::vector< std::pair< _DifferenceType, _DifferenceType > > * __pieces)
```

Sampling based splitting for parallel multiway-merge routine.
References _GLIBCXX_PARALLEL_LENGTH, __gnu_parallel::_Settings::get(), and __gnu_parallel::_Settings::merge_oversampling.

multiway_merge_sentinels()

```
template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _↩
Compare>
_RAIterOut __gnu_parallel::multiway_merge_sentinels (
             _RAIterPairIterator __seqs_begin,
             _RAIterPairIterator __seqs_end,
             _RAIterOut __target,
             _DifferenceTp __length,
             _Compare __comp,
             __gnu_parallel::sequential_tag )
```

Multiway Merge Frontend.
Merge the sequences specified by seqs_begin and __seqs_end into __target. __seqs_begin and __seqs_end must point to a sequence of pairs. These pairs must contain an iterator to the beginning of a sequence in their first entry and an iterator the _M_end of the same sequence in their second entry.
Ties are broken arbitrarily. See stable_multiway_merge for a variant that breaks ties by sequence number but is slower.
The first entries of the pairs (i.e. the begin iterators) will be moved forward accordingly.
The output sequence has to provide enough space for all elements that are written to it.
This function will merge the input sequences:

- not stable

- parallel, depending on the input size and Settings
- using sampling for splitting
- using sentinels

You have to take care that the element the _M_end iterator points to is readable and contains a value that is greater than any other non-sentinel value in all sequences.
Example:

```
int sequences[10][11];
for (int __i = 0; __i < 10; ++__i)
  for (int __j = 0; __i < 11; ++__j)
    sequences[__i][__j] = __j; // __last one is sentinel!

int __out[33];
std::vector<std::pair<int*> > seqs;
for (int __i = 0; __i < 10; ++__i)
  { seqs.push(std::make_pair<int*>(sequences[__i],
                                   sequences[__i] + 10)) }

multiway_merge(seqs.begin(), seqs.end(), __target, std::less<int>(), 33);
```

Precondition

All input sequences must be sorted.

Target must provide enough space to merge out length elements or the number of elements in all sequences, whichever is smaller.

For each __i, __seqs_begin[__i].second must be the end marker of the sequence, but also reference the one more __sentinel element.

Postcondition

[__target, return __value) contains merged __elements from the input sequences.

return __value - __target = min(__length, number of elements in all sequences).

See also

stable_multiway_merge_sentinels

Template Parameters

_RAIterPairIterator	iterator over sequence of pairs of iterators
_RAIterOut	iterator over target sequence
_DifferenceTp	difference type for the sequence
_Compare	strict weak ordering type to compare elements in sequences

Parameters

__seqs_begin	__begin of sequence __sequence
__seqs_end	_M_end of sequence __sequence

__target	target sequence to merge to.
__comp	strict weak ordering to use for element comparison.
__length	Maximum length to merge, possibly larger than the number of elements available.

Returns

_M_end iterator of output sequence

References __sequential_multiway_merge(), and _GLIBCXX_CALL.

parallel_multiway_merge()

```
template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename ↩
_DifferenceTp, typename _Splitter, typename _Compare>
_RAIter3 __gnu_parallel::parallel_multiway_merge (
            _RAIterIterator __seqs_begin,
            _RAIterIterator __seqs_end,
            _RAIter3 __target,
            _Splitter __splitter,
            _DifferenceTp __length,
            _Compare __comp,
            _ThreadIndex __num_threads)
```

Parallel multi-way merge routine.
The _GLIBCXX_PARALLEL_DECISION is based on the branching factor and runtime settings.
Must not be called if the number of sequences is 1.

Template Parameters

_Splitter	functor to split input (either __exact or sampling based)
__stable	Stable merging incurs a performance penalty.
__sentinel	Ignored.

Parameters

__seqs_begin	Begin iterator of iterator pair input sequence.
__seqs_end	End iterator of iterator pair input sequence.
__target	Begin iterator of output sequence.
__comp	Comparator.
__length	Maximum length to merge, possibly larger than the number of elements available.

Returns

End iterator of output sequence.

References __is_sorted(), __sequential_multiway_merge(), _GLIBCXX_CALL, _GLIBCXX_PARALLEL_LENGTH, __gnu_parallel::_Settings::get(), std::make_pair(), __gnu_parallel::_Settings::merge_oversampling, and std::min().
Referenced by __parallel_merge_advance().

parallel_sort_mwms()

```
template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
void __gnu_parallel::parallel_sort_mwms (
            _RAIter __begin,
            _RAIter __end,
            _Compare __comp,
            _ThreadIndex __num_threads)
```

PMWMS main call.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__comp	Comparator.
__num_threads	Number of threads to use.

References _GLIBCXX_CALL, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_num_threads, __gnu_parallel::_PMWMSSortingD
__gnu_parallel::_PMWMSSortingData< _RAIter >::_M_pieces, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_samples,
__gnu_parallel::_PMWMSSortingData< _RAIter >::_M_source, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_starts,
__gnu_parallel::_PMWMSSortingData< _RAIter >::_M_temporary, __gnu_parallel::_Settings::get(), parallel_sort_mwms_pu(),
and __gnu_parallel::_Settings::sort_mwms_oversampling.
Referenced by __parallel_sort(), __parallel_sort(), __parallel_sort(), and __parallel_sort().

parallel_sort_mwms_pu()

```
template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
void __gnu_parallel::parallel_sort_mwms_pu (
            _PMWMSSortingData< _RAIter > * __sd,
            _Compare & __comp)
```

PMWMS code executed by each thread.

Parameters

__sd	Pointer to algorithm data.
__comp	Comparator.

References __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_num_threads, __gnu_parallel::_PMWMSSortingData< _RAIter >::_
__gnu_parallel::_PMWMSSortingData< _RAIter >::_M_source, __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_starts,
__gnu_parallel::_PMWMSSortingData< _RAIter >::_M_temporary, __gnu_parallel::_Settings::get(), std::make_pair(),
__gnu_parallel::_Settings::sort_mwms_oversampling, and std::uninitialized_copy().
Referenced by parallel_sort_mwms().

4.6.5 Variable Documentation

_CASable_bits

```
const int __gnu_parallel::_CASable_bits  [static]
```

Number of bits of _CASable.
Referenced by __decode2(), and __encode2().

_CASable_mask

```
const _CASable __gnu_parallel::_CASable_mask  [static]
```

_CASable with the right half of bits set to 1.
Referenced by __decode2().

4.7 __gnu_pbds Namespace Reference

Classes

- struct associative_tag
- class basic_branch
- struct basic_branch_tag
- class basic_hash_table
- struct basic_hash_tag
- struct basic_invalidation_guarantee
- struct binary_heap_tag
- struct binomial_heap_tag
- class cc_hash_max_collision_check_resize_trigger
- class cc_hash_table
- struct cc_hash_tag
- struct container_error
- struct container_tag
- struct container_traits
- struct container_traits_base
- struct container_traits_base< binary_heap_tag >
- struct container_traits_base< binomial_heap_tag >
- struct container_traits_base< cc_hash_tag >
- struct container_traits_base< gp_hash_tag >
- struct container_traits_base< list_update_tag >
- struct container_traits_base< ov_tree_tag >
- struct container_traits_base< pairing_heap_tag >
- struct container_traits_base< pat_trie_tag >
- struct container_traits_base< rb_tree_tag >
- struct container_traits_base< rc_binomial_heap_tag >
- struct container_traits_base< splay_tree_tag >
- struct container_traits_base< thin_heap_tag >
- class direct_mask_range_hashing
- class direct_mod_range_hashing
- class gp_hash_table
- struct gp_hash_tag
- class hash_exponential_size_policy
- class hash_load_check_resize_trigger
- class hash_prime_size_policy
- class hash_standard_resize_policy
- struct insert_error
- struct join_error
- class linear_probe_fn
- class list_update
- struct list_update_tag
- class lu_counter_policy
- class lu_move_to_front_policy

- struct null_node_update
- struct null_type
- struct ov_tree_tag
- struct pairing_heap_tag
- struct pat_trie_tag
- struct point_invalidation_guarantee
- class priority_queue
- struct priority_queue_tag
- class quadratic_probe_fn
- struct range_invalidation_guarantee
- struct rb_tree_tag
- struct rc_binomial_heap_tag
- struct resize_error
- class sample_probe_fn
- class sample_range_hashing
- class sample_ranged_hash_fn
- class sample_ranged_probe_fn
- class sample_resize_policy
- class sample_resize_trigger
- class sample_size_policy
- class sample_tree_node_update
- struct sample_trie_access_traits
- class sample_trie_node_update
- struct sample_update_policy
- struct sequence_tag
- struct splay_tree_tag
- struct string_tag
- struct thin_heap_tag
- class tree
- class tree_order_statistics_node_update
- struct tree_tag
- class trie
- class trie_order_statistics_node_update
- class trie_prefix_search_node_update
- struct trie_string_access_traits
- struct trie_tag
- struct trivial_iterator_tag

Typedefs

- typedef void trivial_iterator_difference_type

Functions

- void **__throw_container_error** ()
- void **__throw_insert_error** ()
- void **__throw_join_error** ()
- void **__throw_resize_error** ()

Variables

- template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc>
  detail::integral_constant< int, Reverse > **trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::s_rev_ind**

4.7.1 Detailed Description

GNU extensions for policy-based data structures for public use.

4.8 __gnu_sequential Namespace Reference

4.8.1 Detailed Description

GNU sequential classes for public use.

4.9 abi Namespace Reference

4.9.1 Detailed Description

The cross-vendor C++ Application Binary Interface. A namespace alias to __cxxabiv1, but user programs should use the alias 'abi'.
A brief overview of an ABI is given in the libstdc++ FAQ, question 5.8 (you may have a copy of the FAQ locally, or you can view the online version at http://gcc.gnu.org/onlinedocs/libstdc++/faq.html#5_8).
GCC subscribes to a cross-vendor ABI for C++, sometimes called the IA64 ABI because it happens to be the native ABI for that platform. It is summarized at http://www.codesourcery.com/cxx-abi/ along with the current specification.
For users of GCC greater than or equal to 3.x, entry points are available in <cxxabi.h>, which notes, *'It is not normally necessary for user programs to include this header, or use the entry points directly. However, this header is available should that be needed.'*

4.10 std Namespace Reference

Namespaces

- namespace __debug
- namespace __detail
- namespace __parallel
- namespace chrono
- namespace decimal
- namespace experimental
- namespace filesystem
- namespace literals
- namespace placeholders
- namespace regex_constants
- namespace rel_ops
- namespace this_thread
- namespace tr1
- namespace tr2

Classes

- class __basic_future
- class __codecvt_abstract_base
- class __ctype_abstract_base
- struct __is_fast_hash
- struct __is_location_invariant
- class __new_allocator
- struct __numeric_limits_base
- struct _Base_bitset
- struct _Base_bitset< 0 >
- struct _Base_bitset< 1 >
- class _Bind
- class _Bind_result
- class _Deque_base
- struct _Deque_iterator
- class _Function_base
- struct _Fwd_list_base
- struct _Fwd_list_const_iterator
- struct _Fwd_list_iterator
- struct _Fwd_list_node
- struct _Fwd_list_node_base
- class _List_base
- struct _List_const_iterator
- struct _List_iterator
- struct _List_node
- class _Not_fn
- struct _Placeholder
- struct _Sp_ebo_helper< _Tp, false >
- struct _Sp_ebo_helper< _Tp, true >
- class _Temporary_buffer
- struct _Vector_base
- struct add_const
- struct add_cv
- struct add_lvalue_reference
- struct add_pointer
- struct add_rvalue_reference
- struct add_volatile
- struct adopt_lock_t
- struct aligned_storage
- struct aligned_union
- struct alignment_of
- class allocator
- class allocator< void >
- struct allocator_traits
- struct allocator_traits< allocator< _Tp > >
- struct allocator_traits< allocator< void > >
- struct allocator_traits< pmr::polymorphic_allocator< _Tp > >
- struct array
- struct atomic
- struct atomic< _Tp * >

- struct atomic< bool >
- struct atomic< char >
- struct atomic< char16_t >
- struct atomic< char32_t >
- struct atomic< int >
- struct atomic< long >
- struct atomic< long long >
- struct atomic< short >
- struct atomic< signed char >
- struct atomic< unsigned char >
- struct atomic< unsigned int >
- struct atomic< unsigned long >
- struct atomic< unsigned long long >
- struct atomic< unsigned short >
- struct atomic< wchar_t >
- struct atomic_flag
- class auto_ptr
- struct auto_ptr_ref
- class back_insert_iterator
- class bad_alloc
- class bad_cast
- class bad_exception
- class bad_function_call
- class bad_typeid
- class bad_weak_ptr
- class basic_filebuf
- class basic_fstream
- class basic_ifstream
- class basic_ios
- class basic_iostream
- class basic_istream
- class basic_istringstream
- class basic_ofstream
- class basic_ostream
- class basic_ostringstream
- class basic_regex
- class basic_streambuf
- class basic_string
- class basic_string_view
- class basic_stringbuf
- class basic_stringstream
- class bernoulli_distribution
- struct bidirectional_iterator_tag
- struct binary_function
- class binary_negate
- class binder1st
- class binder2nd
- class binomial_distribution
- class bitset
- class cauchy_distribution
- struct char_traits

- struct char_traits< __gnu_cxx::character< _Value, _Int, _St > >
- struct char_traits< char >
- struct char_traits< wchar_t >
- class chi_squared_distribution
- class codecvt
- class codecvt< _InternT, _ExternT, encoding_state >
- class codecvt< char, char, mbstate_t >
- class codecvt< char16_t, char, mbstate_t >
- class codecvt< char32_t, char, mbstate_t >
- class codecvt< wchar_t, char, mbstate_t >
- class codecvt_base
- class codecvt_byname
- class common_iterator
- struct common_type
- struct common_type< chrono::duration< _Rep, _Period > >
- struct common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >
- struct common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >
- struct common_type< chrono::time_point< _Clock, _Duration > >
- struct common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >
- struct common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >
- struct compare_three_way_result
- class complex
- class complex< double >
- class complex< float >
- class complex< long double >
- class condition_variable
- class condition_variable_any
- struct conditional
- class const_mem_fun1_ref_t
- class const_mem_fun1_t
- class const_mem_fun_ref_t
- class const_mem_fun_t
- struct contiguous_iterator_tag
- class copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>
- class counted_iterator
- class ctype
- class ctype< char >
- class ctype< wchar_t >
- struct ctype_base
- class ctype_byname
- class ctype_byname< char >
- struct default_delete
- struct default_delete< _Tp[ ]>
- struct default_sentinel_t
- struct defer_lock_t
- class deque
- struct destroying_delete_t
- class discard_block_engine
- class discrete_distribution
- struct divides
- struct divides< void >

- class domain_error
- struct enable_if
- class enable_shared_from_this
- class encoding_state
- struct equal_to
- struct equal_to< void >
- class error_category
- class error_code
- class error_condition
- class exception
- class exception_ptr
- class exponential_distribution
- struct extent
- class extreme_value_distribution
- class fisher_f_distribution
- struct forward_iterator_tag
- class forward_list
- class fpos
- struct from_chars_result
- class front_insert_iterator
- class function< _Res(_ArgTypes...)>
- class function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>
- class future
- class future< _Res & >
- class future< void >
- class future_error
- class gamma_distribution
- class geometric_distribution
- struct greater
- struct greater< void >
- struct greater_equal
- struct greater_equal< void >
- class gslice
- class gslice_array
- struct has_virtual_destructor
- struct hash
- struct hash< __debug::bitset< _Nb > >
- struct hash< __debug::vector< bool, _Alloc > >
- struct hash< __gnu_cxx::__u16vstring >
- struct hash< __gnu_cxx::__u32vstring >
- struct hash< __gnu_cxx::__vstring >
- struct hash< __gnu_cxx::__wvstring >
- struct hash< __gnu_cxx::throw_value_limit >
- struct hash< __gnu_cxx::throw_value_random >
- struct hash< __gnu_debug::basic_string< _CharT > >
- struct hash< __shared_ptr< _Tp, _Lp > >
- struct hash< _Tp * >
- struct hash< basic_string< char, char_traits< char >, _Alloc > >
- struct hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > >
- struct hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > >
- struct hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > >

- struct hash< bool >
- struct hash< char >
- struct hash< char16_t >
- struct hash< char32_t >
- struct hash< double >
- struct hash< error_code >
- struct hash< error_condition >
- struct hash< experimental::optional< _Tp > >
- struct hash< experimental::shared_ptr< _Tp > >
- struct hash< float >
- struct hash< int >
- struct hash< long >
- struct hash< long double >
- struct hash< long long >
- struct hash< shared_ptr< _Tp > >
- struct hash< short >
- struct hash< signed char >
- struct hash< thread::id >
- struct hash< type_index >
- struct hash< unique_ptr< _Tp, _Dp > >
- struct hash< unsigned char >
- struct hash< unsigned int >
- struct hash< unsigned long >
- struct hash< unsigned long long >
- struct hash< unsigned short >
- struct hash< wchar_t >
- struct hash<::bitset< _Nb > >
- struct hash<::vector< bool, _Alloc > >
- struct identity
- class independent_bits_engine
- class indirect_array
- class initializer_list
- class inout_ptr_t
- struct input_iterator_tag
- class insert_iterator
- struct integer_sequence
- struct integral_constant
- class invalid_argument
- class ios_base
- struct is_abstract
- struct is_arithmetic
- struct is_array
- struct is_assignable
- struct is_base_of
- struct is_bind_expression
- struct is_bind_expression< _Bind< _Signature > >
- struct is_bind_expression< _Bind_result< _Result, _Signature > >
- struct is_bind_expression< const _Bind< _Signature > >
- struct is_bind_expression< const _Bind_result< _Result, _Signature > >
- struct is_bind_expression< const volatile _Bind< _Signature > >
- struct is_bind_expression< const volatile _Bind_result< _Result, _Signature > >

- struct is_bind_expression< volatile _Bind< _Signature > >
- struct is_bind_expression< volatile _Bind_result< _Result, _Signature > >
- struct is_class
- struct is_compound
- struct is_const
- struct is_constructible
- struct is_copy_assignable
- struct is_copy_constructible
- struct is_default_constructible
- struct is_destructible
- struct is_empty
- struct is_enum
- struct is_error_code_enum
- struct is_error_code_enum< future_errc >
- struct is_error_condition_enum
- struct is_floating_point
- struct is_function
- struct is_fundamental
- struct is_integral
- struct is_layout_compatible
- struct is_literal_type
- struct is_lvalue_reference
- struct is_member_function_pointer
- struct is_member_object_pointer
- struct is_member_pointer
- struct is_move_assignable
- struct is_move_constructible
- struct is_nothrow_assignable
- struct is_nothrow_constructible
- struct is_nothrow_copy_assignable
- struct is_nothrow_copy_constructible
- struct is_nothrow_default_constructible
- struct is_nothrow_destructible
- struct is_nothrow_move_assignable
- struct is_nothrow_move_constructible
- struct is_object
- struct is_placeholder
- struct is_placeholder< _Placeholder< _Num > >
- struct is_pod
- struct is_pointer
- struct is_pointer_interconvertible_base_of
- struct is_polymorphic
- struct is_reference
- struct is_rvalue_reference
- struct is_same
- struct is_scalar
- struct is_signed
- struct is_standard_layout
- struct is_trivial
- struct is_trivially_assignable
- struct is_trivially_constructible

- struct is_trivially_copy_assignable
- struct is_trivially_copy_constructible
- struct is_trivially_copyable
- struct is_trivially_default_constructible
- struct is_trivially_destructible
- struct is_trivially_move_assignable
- struct is_trivially_move_constructible
- struct is_union
- struct is_unsigned
- struct is_void
- struct is_volatile
- class istream_iterator
- class istreambuf_iterator
- struct iterator
- struct iterator_traits
- struct iterator_traits< _Tp * >
- class length_error
- struct less
- struct less< void >
- struct less_equal
- struct less_equal< void >
- class linear_congruential_engine
- class list
- class locale
- class lock_guard
- class logic_error
- struct logical_and
- struct logical_and< void >
- struct logical_not
- struct logical_not< void >
- struct logical_or
- struct logical_or< void >
- class lognormal_distribution
- struct make_signed
- struct make_unsigned
- class map
- class mask_array
- class match_results
- class mem_fun1_ref_t
- class mem_fun1_t
- class mem_fun_ref_t
- class mem_fun_t
- class mersenne_twister_engine
- class messages
- struct messages_base
- class messages_byname
- struct minus
- struct minus< void >
- struct modulus
- struct modulus< void >
- class money_base

- class money_get
- class money_put
- class moneypunct
- class moneypunct_byname
- class move_iterator
- class move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>
- class move_sentinel
- class multimap
- struct multiplies
- struct multiplies< void >
- class multiset
- class mutex
- struct negate
- struct negate< void >
- class negative_binomial_distribution
- class nested_exception
- class normal_distribution
- struct not_equal_to
- struct not_equal_to< void >
- class num_get
- class num_put
- struct numeric_limits
- struct numeric_limits< bool >
- struct numeric_limits< char >
- struct numeric_limits< char16_t >
- struct numeric_limits< char32_t >
- struct numeric_limits< double >
- struct numeric_limits< float >
- struct numeric_limits< int >
- struct numeric_limits< long >
- struct numeric_limits< long double >
- struct numeric_limits< long long >
- struct numeric_limits< short >
- struct numeric_limits< signed char >
- struct numeric_limits< unsigned char >
- struct numeric_limits< unsigned int >
- struct numeric_limits< unsigned long >
- struct numeric_limits< unsigned long long >
- struct numeric_limits< unsigned short >
- struct numeric_limits< wchar_t >
- class numpunct
- class numpunct_byname
- struct once_flag
- class ostream_iterator
- class ostreambuf_iterator
- class out_of_range
- class out_ptr_t
- struct output_iterator_tag
- class overflow_error
- struct owner_less
- struct owner_less< shared_ptr< _Tp > >

- struct owner_less< void >
- struct owner_less< weak_ptr< _Tp > >
- class packaged_task< _Res(_ArgTypes...)>
- struct pair
- class philox_engine
- class piecewise_constant_distribution
- struct piecewise_construct_t
- class piecewise_linear_distribution
- struct plus
- class pointer_to_binary_function
- class pointer_to_unary_function
- struct pointer_traits
- struct pointer_traits< _Tp * >
- class poisson_distribution
- class priority_queue
- class promise
- class promise< _Res & >
- class promise< void >
- class queue
- struct random_access_iterator_tag
- class random_device
- class range_error
- struct rank
- struct ratio
- struct ratio_equal
- struct ratio_greater
- struct ratio_greater_equal
- struct ratio_less
- struct ratio_less_equal
- struct ratio_not_equal
- class raw_storage_iterator
- class recursive_mutex
- class recursive_timed_mutex
- class reference_wrapper
- class regex_error
- class regex_iterator
- class regex_token_iterator
- class regex_traits
- struct remove_all_extents
- struct remove_const
- struct remove_cv
- struct remove_extent
- struct remove_pointer
- struct remove_reference
- struct remove_volatile
- struct result_of
- class reverse_iterator
- class runtime_error
- class scoped_allocator_adaptor
- class seed_seq
- class set

- class shared_future
- class shared_future< _Res & >
- class shared_future< void >
- class shared_lock
- class shared_ptr
- class shared_timed_mutex
- class shuffle_order_engine
- class slice
- class slice_array
- class stack
- class student_t_distribution
- class sub_match
- class subtract_with_carry_engine
- class system_error
- class thread
- class time_base
- class time_get
- class time_get_byname
- class time_put
- class time_put_byname
- class timed_mutex
- struct to_chars_result
- struct try_to_lock_t
- class tuple
- class tuple< _T1, _T2 >
- struct tuple_element
- struct tuple_element< 0, pair< _Tp1, _Tp2 > >
- struct tuple_element< 1, pair< _Tp1, _Tp2 > >
- struct tuple_element< __i, tuple< _Types... > >
- struct tuple_element< _Ind, array< _Tp, _Nm > >
- struct tuple_size
- struct tuple_size< array< _Tp, _Nm > >
- struct tuple_size< pair< _Tp1, _Tp2 > >
- struct tuple_size< tuple< _Elements... > >
- struct type_index
- class type_info
- struct type_order
- struct unary_function
- class unary_negate
- class underflow_error
- struct underlying_type
- class uniform_int_distribution
- class uniform_real_distribution
- class unique_lock
- class unique_ptr
- class unique_ptr< _Tp[ ], _Dp >
- class unordered_map
- class unordered_multimap
- class unordered_multiset
- class unordered_set
- struct uses_allocator

- struct uses_allocator< tuple< _Types... >, _Alloc >
- class valarray
- class vector
- class vector< bool, _Alloc >
- class wbuffer_convert
- class weak_ptr
- class weibull_distribution
- class wstring_convert

Concepts

- concept assignable_from
- concept common_reference_with
- concept common_with
- concept constructible_from
- concept convertible_to
- concept copy_constructible
- concept default_initializable
- concept derived_from
- concept destructible
- concept equivalence_relation
- concept indirectly_comparable
- concept indirectly_copyable
- concept indirectly_movable
- concept indirectly_readable
- concept indirectly_swappable
- concept indirectly_writable
- concept invocable
- concept mergeable
- concept move_constructible
- concept permutable
- concept predicate
- concept regular_invocable
- concept relation
- concept same_as
- concept sortable
- concept strict_weak_order
- concept uniform_random_bit_generator
- concept weakly_incrementable

Typedefs

- template<typename _Tp>
  using __allocator_base
- typedef unsigned char **__atomic_flag_data_type**
- typedef FILE **__c_file**
- typedef __locale_t **__c_locale**
- typedef __gthread_mutex_t **__c_lock**
- template<bool _Cond, typename _If, typename _Else>
  using **__conditional_t**

- template<typename _Tp, typename _Up = typename remove_cv<_Tp>::type, typename = typename enable_if<is_same<_Tp, _Up>::value>::type, size_t = tuple_size<_Tp>::value>
  using **__enable_if_has_tuple_size**
- template<bool _Cond, typename _Tp = void>
  using **__enable_if_t**
- template<typename _Fn, typename _Op>
  using **__function_guide_t**
- template<typename _Func, typename _SfinaeType>
  using **__has_is_transparent_t**
- template<typename _Tp>
  using **__indirect_value_t**
- template<typename _ToElementType, typename _FromElementType>
  using **__is_array_convertible**
- template<typename _Tp, typename _Tp2 = typename decay<_Tp>::type>
  using **__is_socketlike**
- template<typename _InputIterator>
  using **__iter_key_t**
- template<typename _InputIterator>
  using **__iter_to_alloc_t**
- template<typename _InputIterator>
  using **__iter_val_t**
- template<typename _Tp, typename _Up>
  using **__like_t**
- template<typename _Ptr, typename _Tp>
  using __ptr_rebind
- template<typename _Is, typename _Tp>
  using **__rvalue_stream_extraction_t**
- template<typename _Os, typename _Tp>
  using **__rvalue_stream_insertion_t**
- typedef basic_string< char > **__sso_string**
- template<size_t __i, typename _Tp>
  using **__tuple_element_t**
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>
  using **__umap_hashtable**
- template<bool _Cache>
  using __umap_traits
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>
  using **__ummap_hashtable**
- template<bool _Cache>
  using __ummap_traits
- template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>
  using **__umset_hashtable**
- template<bool _Cache>
  using __umset_traits
- template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>
  using **__uset_hashtable**
- template<bool _Cache>
  using __uset_traits

- template<typename _Fn, typename... _Args>
  using **_Bind_back_t**
- template<typename _Fn, typename... _Args>
  using **_Bind_front_t**
- typedef unsigned long **_Bit_type**
- template<typename _Path, typename _Result = _Path, typename _Path2 = decltype(std::declval<_Path&>().make_preferred().filename())>
  using **_If_fs_path**
- template<typename _Tp>
  using add_lvalue_reference_t
- template<typename _Tp>
  using add_pointer_t
- template<typename _Tp>
  using add_rvalue_reference_t
- template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
  using aligned_storage_t
- template<size_t _Len, typename... _Types>
  using **aligned_union_t**
- typedef atomic< bool > atomic_bool
- typedef atomic< char > atomic_char
- typedef atomic< char16_t > atomic_char16_t
- typedef atomic< char32_t > atomic_char32_t
- typedef atomic< int > atomic_int
- typedef atomic< int16_t > atomic_int16_t
- typedef atomic< int32_t > atomic_int32_t
- typedef atomic< int64_t > atomic_int64_t
- typedef atomic< int8_t > atomic_int8_t
- typedef atomic< int_fast16_t > atomic_int_fast16_t
- typedef atomic< int_fast32_t > atomic_int_fast32_t
- typedef atomic< int_fast64_t > atomic_int_fast64_t
- typedef atomic< int_fast8_t > atomic_int_fast8_t
- typedef atomic< int_least16_t > atomic_int_least16_t
- typedef atomic< int_least32_t > atomic_int_least32_t
- typedef atomic< int_least64_t > atomic_int_least64_t
- typedef atomic< int_least8_t > atomic_int_least8_t
- typedef atomic< intmax_t > atomic_intmax_t
- typedef atomic< intptr_t > atomic_intptr_t
- typedef atomic< long long > atomic_llong
- typedef atomic< long > atomic_long
- typedef atomic< ptrdiff_t > atomic_ptrdiff_t
- typedef atomic< signed char > atomic_schar
- typedef atomic< short > atomic_short
- typedef atomic< size_t > atomic_size_t
- typedef atomic< unsigned char > atomic_uchar
- typedef atomic< unsigned int > atomic_uint
- typedef atomic< uint16_t > atomic_uint16_t
- typedef atomic< uint32_t > atomic_uint32_t
- typedef atomic< uint64_t > atomic_uint64_t
- typedef atomic< uint8_t > atomic_uint8_t
- typedef atomic< uint_fast16_t > atomic_uint_fast16_t
- typedef atomic< uint_fast32_t > atomic_uint_fast32_t
- typedef atomic< uint_fast64_t > atomic_uint_fast64_t

- typedef atomic< uint_fast8_t > atomic_uint_fast8_t
- typedef atomic< uint_least16_t > atomic_uint_least16_t
- typedef atomic< uint_least32_t > atomic_uint_least32_t
- typedef atomic< uint_least64_t > atomic_uint_least64_t
- typedef atomic< uint_least8_t > atomic_uint_least8_t
- typedef atomic< uintmax_t > atomic_uintmax_t
- typedef atomic< uintptr_t > atomic_uintptr_t
- typedef atomic< unsigned long long > atomic_ullong
- typedef atomic< unsigned long > atomic_ulong
- typedef atomic< unsigned short > atomic_ushort
- typedef atomic< wchar_t > atomic_wchar_t
- using **atto**
- using **centi**
- typedef match_results< const char * > **cmatch**
- template<typename... _Ts>
  using **common_comparison_category_t**
- template<typename... _Tp>
  using common_type_t
- template<typename _Tp, typename _Up = _Tp>
  using compare_three_way_result_t
- template<bool _Cond, typename _Iftrue, typename _Iffalse>
  using conditional_t
- template<input_iterator _It>
  using **const_iterator**
- template<semiregular _Sent>
  using **const_sentinel**
- typedef regex_iterator< const char * > **cregex_iterator**
- typedef regex_token_iterator< const char * > cregex_token_iterator
- typedef sub_match< const char * > csub_match
- using **deca**
- template<typename _Tp>
  using decay_t
- using **deci**
- typedef minstd_rand0 **default_random_engine**
- template<typename _IndexType, size_t _Rank>
  using **dextents**
- template<size_t _Rank, typename _IndexType = size_t>
  using **dims**
- template<bool _Cond, typename _Tp = void>
  using enable_if_t
- using **exa**
- using false_type
- using **femto**
- typedef basic_filebuf< char > filebuf
- typedef basic_fstream< char > fstream
- using **giga**
- using **hecto**
- typedef basic_ifstream< char > ifstream
- template<size_t... _Idx>
  using index_sequence
- template<typename... _Types>
  using index_sequence_for

- template<typename _Fn, typename... _Is>
  using **indirect_result_t**
- typedef basic_ios< char > ios
- typedef basic_iostream< char > iostream
- typedef basic_istream< char > istream
- typedef basic_istringstream< char > istringstream
- template<indirectly_readable _Tp>
  using **iter_common_reference_t**
- template<indirectly_readable _It>
  using **iter_const_reference_t**
- template<typename _Tp>
  using **iter_difference_t**
- template<__detail::__dereferenceable _Tp>
  using **iter_reference_t**
- template<__detail::__dereferenceable _Tp>
  using iter_rvalue_reference_t
- template<typename _Tp>
  using **iter_value_t**
- using **kilo**
- typedef shuffle_order_engine< minstd_rand0, 256 > **knuth_b**
- template<size_t _Num>
  using make_index_sequence
- template<typename _Tp, _Tp _Num>
  using make_integer_sequence
- template<typename _Tp>
  using make_signed_t
- template<typename _Tp>
  using make_unsigned_t
- using **mega**
- using **micro**
- using **milli**
- typedef linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > minstd_rand
- typedef linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > minstd_rand0
- typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > mt19937
- typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67fffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > mt19937_64
- using **nano**
- typedef void(* new_handler) ()
- typedef decltype(nullptr) **nullptr_t**
- typedef basic_ofstream< char > ofstream
- typedef basic_ostream< char > ostream
- typedef basic_ostringstream< char > ostringstream
- using **peta**
- typedef philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 > philox4x32
- typedef philox_engine< uint_fast64_t, 64, 4, 10, 0xCA5A826395121157, 0x9E3779B97F4A7C15, 0xD2E7470↩EE14C6C93, 0xBB67AE8584CAA73B > philox4x64
- using **pico**
- template<indirectly_readable _Iter, indirectly_regular_unary_invocable< _Iter > _Proj>
  using projected

- template<indirectly_readable _Iter, indirectly_regular_unary_invocable< _Iter > _Proj>
  using **projected_value_t**
- typedef __PTRDIFF_TYPE__ **ptrdiff_t**
- typedef discard_block_engine< ranlux24_base, 223, 23 > **ranlux24**
- typedef subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 > **ranlux24_base**
- typedef discard_block_engine< ranlux48_base, 389, 11 > **ranlux48**
- typedef subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 > **ranlux48_base**
- template<typename _R1, typename _R2>
  using ratio_add
- template<typename _R1, typename _R2>
  using ratio_divide
- template<typename _R1, typename _R2>
  using ratio_multiply
- template<typename _R1, typename _R2>
  using ratio_subtract
- typedef basic_regex< char > regex
- template<typename _Tp>
  using remove_all_extents_t
- template<typename _Tp>
  using remove_extent_t
- template<typename _Tp>
  using remove_pointer_t
- template<typename _Tp>
  using remove_reference_t
- template<typename _Tp>
  using result_of_t
- typedef __SIZE_TYPE__ **size_t**
- typedef match_results< string::const_iterator > **smatch**
- typedef regex_iterator< string::const_iterator > **sregex_iterator**
- typedef regex_token_iterator< string::const_iterator > sregex_token_iterator
- typedef sub_match< string::const_iterator > ssub_match
- typedef basic_streambuf< char > streambuf
- typedef long long streamoff
- typedef fpos< mbstate_t > streampos
- typedef ptrdiff_t streamsize
- typedef basic_string< char > string
- using **string_view**
- typedef basic_stringbuf< char > stringbuf
- typedef basic_stringstream< char > stringstream
- using **tera**
- typedef void(* terminate_handler) ()
- using true_type
- template<size_t __i, typename _Tp>
  using **tuple_element_t**
- typedef fpos< mbstate_t > u16streampos
- typedef basic_string< char16_t > u16string
- using **u16string_view**
- typedef fpos< mbstate_t > u32streampos
- typedef basic_string< char32_t > u32string
- using **u32string_view**
- template<typename _Tp>
  using underlying_type_t

- typedef match_results< const wchar_t * > **wcmatch**
- typedef regex_iterator< const wchar_t * > **wcregex_iterator**
- typedef regex_token_iterator< const wchar_t * > wcregex_token_iterator
- typedef sub_match< const wchar_t * > wcsub_match
- typedef basic_filebuf< wchar_t > wfilebuf
- typedef basic_fstream< wchar_t > wfstream
- typedef basic_ifstream< wchar_t > wifstream
- typedef basic_ios< wchar_t > wios
- typedef basic_iostream< wchar_t > wiostream
- typedef basic_istream< wchar_t > wistream
- typedef basic_istringstream< wchar_t > wistringstream
- typedef basic_ofstream< wchar_t > wofstream
- typedef basic_ostream< wchar_t > wostream
- typedef basic_ostringstream< wchar_t > wostringstream
- typedef basic_regex< wchar_t > wregex
- typedef match_results< wstring::const_iterator > **wsmatch**
- typedef regex_iterator< wstring::const_iterator > **wsregex_iterator**
- typedef regex_token_iterator< wstring::const_iterator > wsregex_token_iterator
- typedef sub_match< wstring::const_iterator > wssub_match
- typedef basic_streambuf< wchar_t > wstreambuf
- typedef fpos< mbstate_t > wstreampos
- typedef basic_string< wchar_t > wstring
- using **wstring_view**
- typedef basic_stringbuf< wchar_t > wstringbuf
- typedef basic_stringstream< wchar_t > wstringstream

Enumerations

- enum { **_S_word_bit** }
- enum **_Ios_Fmtflags** {
  _S_boolalpha , **_S_dec** , **_S_fixed** , **_S_hex** ,
  _S_internal , **_S_left** , **_S_oct** , **_S_right** ,
  _S_scientific , **_S_showbase** , **_S_showpoint** , **_S_showpos** ,
  _S_skipws , **_S_unitbuf** , **_S_uppercase** , **_S_adjustfield** ,
  _S_basefield , **_S_floatfield** , **_S_ios_fmtflags_end** , **_S_ios_fmtflags_max** ,
  _S_ios_fmtflags_min }
- enum **_Ios_Iostate** {
  _S_goodbit , **_S_badbit** , **_S_eofbit** , **_S_failbit** ,
  _S_ios_iostate_end , **_S_ios_iostate_max** , **_S_ios_iostate_min** }
- enum **_Ios_Openmode** {
  _S_app , **_S_ate** , **_S_bin** , **_S_in** ,
  _S_out , **_S_trunc** , **_S_noreplace** , **_S_ios_openmode_end** ,
  _S_ios_openmode_max , **_S_ios_openmode_min** }
- enum **_Ios_Seekdir** { **_S_beg** , **_S_cur** , **_S_end** , **_S_ios_seekdir_end** }
- enum **_Lock_policy**
- enum **_Manager_operation** { **__get_type_info** , **__get_functor_ptr** , **__clone_functor** , **__destroy_functor** }
- enum **_Rb_tree_color** { **_S_red** , **_S_black** }
- enum class **align_val_t** : size_t
- enum class chars_format { **scientific** , **fixed** , **hex** , **general** }
- enum **codecvt_mode** { **consume_header** , **generate_header** , **little_endian** }
- enum class cv_status { **no_timeout** , **timeout** }

- enum class **errc** {
  address_family_not_supported , **address_in_use** , **address_not_available** , **already_connected** ,
  argument_list_too_long , **argument_out_of_domain** , **bad_address** , **bad_file_descriptor** ,
  broken_pipe , **connection_aborted** , **connection_already_in_progress** , **connection_refused** ,
  connection_reset , **cross_device_link** , **destination_address_required** , **device_or_resource_busy** ,
  directory_not_empty , **executable_format_error** , **file_exists** , **file_too_large** ,
  filename_too_long , **function_not_supported** , **host_unreachable** , **illegal_byte_sequence** ,
  inappropriate_io_control_operation , **interrupted** , **invalid_argument** , **invalid_seek** ,
  io_error , **is_a_directory** , **message_size** , **network_down** ,
  network_reset , **network_unreachable** , **no_buffer_space** , **no_child_process** ,
  no_lock_available , **no_message** , **no_protocol_option** , **no_space_on_device** ,
  no_such_device_or_address , **no_such_device** , **no_such_file_or_directory** , **no_such_process** ,
  not_a_directory , **not_a_socket** , **not_connected** , **not_enough_memory** ,
  operation_in_progress , **operation_not_permitted** , **operation_not_supported** , **operation_would_block** ,
  permission_denied , **protocol_not_supported** , **read_only_file_system** , **resource_deadlock_would_occur** ,
  resource_unavailable_try_again , **result_out_of_range** , **timed_out** , **too_many_files_open_in_system** ,
  too_many_files_open , **too_many_links** , **too_many_symbolic_link_levels** , **wrong_protocol_type** }
- enum float_denorm_style { denorm_indeterminate , denorm_absent , denorm_present }
- enum float_round_style {
  round_indeterminate , round_toward_zero , round_to_nearest , round_toward_infinity ,
  round_toward_neg_infinity }
- enum class future_errc { **future_already_retrieved** , **promise_already_satisfied** , **no_state** , **broken_promise** }
- enum class future_status { **ready** , **timeout** , **deferred** }
- enum class io_errc { **stream** }
- enum class launch { **async** , **deferred** }
- enum class memory_order : int {
  relaxed , **consume** , **acquire** , **release** ,
  acq_rel , **seq_cst** }

Functions

- template<typename _CharT>
  _CharT * **__add_grouping** (_CharT *__s, _CharT __sep, const char *__gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)
- template<typename _Tp>
  constexpr _Tp * __addressof (_Tp &__r) noexcept
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator **__adjacent_find** (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>
  constexpr void **__adjust_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, _Compare __comp)
- template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename _Alloc, typename... _Args>
  __shared_ptr< _Tp, _Lp > **__allocate_shared** (const _Alloc &__a, _Args &&... __args)
- template<bool _IsMove, typename _OutIter, typename _InIter>
  constexpr void **__assign_one** (_OutIter &__out, _InIter &__in)
- template<typename _Facet>
  const _Facet & **__check_facet** (const _Facet *__f)
- template<typename _Tp>
  _Tp **__complex_abs** (const complex< _Tp > &__z)
- __complex__ double **__complex_acos** (__complex__ double __z)

- __complex__ float **__complex_acos** (__complex__ float __z)
- __complex__ long double **__complex_acos** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_acos** (const std::complex< _Tp > &__z)
- __complex__ double **__complex_acosh** (__complex__ double __z)
- __complex__ float **__complex_acosh** (__complex__ float __z)
- __complex__ long double **__complex_acosh** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_acosh** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  _Tp **__complex_arg** (const complex< _Tp > &__z)
- __complex__ double **__complex_asin** (__complex__ double __z)
- __complex__ float **__complex_asin** (__complex__ float __z)
- __complex__ long double **__complex_asin** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_asin** (const std::complex< _Tp > &__z)
- __complex__ double **__complex_asinh** (__complex__ double __z)
- __complex__ float **__complex_asinh** (__complex__ float __z)
- __complex__ long double **__complex_asinh** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_asinh** (const std::complex< _Tp > &__z)
- __complex__ double **__complex_atan** (__complex__ double __z)
- __complex__ float **__complex_atan** (__complex__ float __z)
- __complex__ long double **__complex_atan** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_atan** (const std::complex< _Tp > &__z)
- __complex__ double **__complex_atanh** (__complex__ double __z)
- __complex__ float **__complex_atanh** (__complex__ float __z)
- __complex__ long double **__complex_atanh** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_atanh** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_cos** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_cosh** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_exp** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_log** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_pow** (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  complex< _Tp > **__complex_pow_unsigned** (complex< _Tp > __x, unsigned __n)
- complex< double > **__complex_proj** (const complex< double > &__z)
- complex< float > **__complex_proj** (const complex< float > &__z)
- complex< long double > **__complex_proj** (const complex< long double > &__z)
- template<typename _Tp>
  std::complex< _Tp > **__complex_proj** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_sin** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_sinh** (const complex< _Tp > &__z)

- template<typename _Tp>
  complex< _Tp > **__complex_sqrt** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_tan** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **__complex_tanh** (const complex< _Tp > &__z)
- int **__convert_from_v** (const __c_locale &__cloc, char *__out, const int __size, const char *__fmt,...)
- template<typename _Tp>
  void **__convert_to_v** (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()
- template<> void **__convert_to_v** (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()
- template<> void **__convert_to_v** (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()
- template<> void **__convert_to_v** (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()
- template<bool _IsMove, typename _II, typename _OI>
  constexpr _OI **__copy_move_a** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>
  constexpr __gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **__copy_move_a** (_II, _II, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &)
- template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>
  constexpr ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > **__copy_move_a** (const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > &)
- template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>
  constexpr _OI **__copy_move_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, _OI)
- template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **__copy_move_a1** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **__copy_move_a1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Tp>
  __gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value,::_Deque_iterator< _Tp, _Tp &, _Tp * > >::__type **__copy_move_a1** (_II __first, _II __last, ::_Deque_iterator< _Tp, _Tp &, _Tp * > __result)
- template<bool _IsMove, typename _II, typename _OI>
  constexpr _OI **__copy_move_a1** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type **__copy_move_a2** (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **__copy_move_a2** (_CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _InIter, typename _Sent, typename _OutIter>
  constexpr _OutIter **__copy_move_a2** (_InIter __first, _Sent __last, _OutIter __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type **__copy_move_a2** (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **__copy_move_a2** (const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result)

- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **__copy_move_a2** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT *__result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value,::_Deque_iterator< _CharT, _CharT &, _CharT * > >::__type **__copy_move_a2** (istreambuf_iterator< _CharT, char_traits< _CharT > > __first, istreambuf_iterator< _CharT, char_traits< _CharT > > __last, ::_Deque_iterator< _CharT, _CharT &, _CharT * > __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **__copy_move_a2** (istreambuf_iterator< _CharT, char_traits< _CharT > >, istreambuf_iterator< _CharT, char_traits< _CharT > >, _CharT *)
- template<bool _IsMove, typename _II, typename _OI>
  constexpr _OI **__copy_move_backward_a** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>
  constexpr __gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **__copy_move_backward_a** (_II, _II, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &)
- template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>
  constexpr ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > **__copy_move_backward_a** (const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > &)
- template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>
  constexpr _OI **__copy_move_backward_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, _OI)
- template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **__copy_move_backward_a1** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **__copy_move_backward_a1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _BI1, typename _BI2>
  constexpr _BI2 **__copy_move_backward_a1** (_BI1 __first, _BI1 __last, _BI2 __result)
- template<bool _IsMove, typename _II, typename _Tp>
  __gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value,::_Deque_iterator< _Tp, _Tp &, _Tp * > >::__type **__copy_move_backward_a1** (_II __first, _II __last, ::_Deque_iterator< _Tp, _Tp &, _Tp * > __result)
- template<bool _IsMove, typename _BI1, typename _BI2>
  constexpr _BI2 **__copy_move_backward_a2** (_BI1 __first, _BI1 __last, _BI2 __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **__copy_move_backward_dit** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **__copy_move_dit** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  constexpr _OutputIterator **__copy_n_a** (_InputIterator __first, _Size __n, _OutputIterator __result, bool)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **__copy_n_a** (istreambuf_iterator< _CharT > __it, _Size __n, _CharT *__result, bool __strict)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value,::_Deque_iterator< _CharT, _CharT &, _CharT * > >::__type **__copy_n_a** (istreambuf_iterator< _CharT, char_traits< _CharT > > __it, _Size __size, ::_Deque_iterator< _CharT, _CharT &, _CharT * > __result, bool __strict)

- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **__copy_n_a** (istreambuf_iterator< _CharT, char_traits< _CharT > >, _Size, _CharT *, bool)
- template<typename _CharT, typename _Traits>
  streamsize **__copy_streambufs** (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT, _Traits > *__sbout)
- template<typename _CharT, typename _Traits>
  streamsize **__copy_streambufs_eof** (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT, _Traits > *__sbout, bool &__ineof)
- template<> streamsize **__copy_streambufs_eof** (basic_streambuf< char > *__sbin, basic_streambuf< char > *__sbout, bool &__ineof)
- template<> streamsize **__copy_streambufs_eof** (basic_streambuf< wchar_t > *__sbin, basic_streambuf< wchar_t > *__sbout, bool &__ineof)
- template<typename _InputIterator, typename _Predicate>
  constexpr iterator_traits< _InputIterator >::difference_type **__count_if** (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- constexpr size_t **__deque_buf_size** (size_t __size)
- template<typename _Tp>
  ptrdiff_t **__distance** (::_List_const_iterator< _Tp >, ::_List_const_iterator< _Tp >, input_iterator_tag)
- template<typename _Tp>
  ptrdiff_t **__distance** (::_List_iterator< _Tp >, ::_List_iterator< _Tp >, input_iterator_tag)
- template<bool _Const, typename _Ptr>
  ptrdiff_t **__distance** (__list::_Iterator< _Const, _Ptr > __first, __list::_Iterator< _Const, _Ptr > __last, input_iterator_tag __tag)
- template<typename _InputIterator>
  constexpr iterator_traits< _InputIterator >::difference_type **__distance** (_InputIterator __first, _InputIterator __last, input_iterator_tag)
- template<typename _OutputIterator>
  void **__distance** (_OutputIterator, _OutputIterator, output_iterator_tag)=delete
- template<typename _RandomAccessIterator>
  constexpr iterator_traits< _RandomAccessIterator >::difference_type **__distance** (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _II1, typename _II2>
  constexpr bool **__equal4** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1, typename _II2, typename _BinaryPredicate>
  constexpr bool **__equal4** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _BinaryPredicate __binary_pred)
- template<typename _II1, typename _II2>
  constexpr bool **__equal_aux** (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _II1, typename _II2, typename _Seq2, typename _Cat2>
  constexpr bool **__equal_aux** (_II1, _II1, const ::__gnu_debug::_Safe_iterator< _II2, _Seq2, _Cat2 > &)
- template<typename _II1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **__equal_aux** (const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, _II2)
- template<typename _II1, typename _Seq1, typename _Cat1, typename _II2, typename _Seq2, typename _Cat2>
  constexpr bool **__equal_aux** (const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II2, _Seq2, _Cat2 > &)
- template<typename _Tp, typename _Ref, typename _Ptr, typename _II>
  __gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value, bool >::__type **__equal_aux1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **__equal_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)

- template<typename _II, typename _Tp, typename _Ref, typename _Ptr>
  __gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value, bool >::__type **__equal_aux1** (_II ↩
  __first1, _II __last1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __first2)
- template<typename _II1, typename _II2>
  constexpr bool **__equal_aux1** (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _Tp, typename _Ref, typename _Ptr, typename _II>
  bool **__equal_dit** (const ::_Deque_iterator< _Tp, _Ref, _Ptr > &__first1, const ::_Deque_iterator< _Tp, _Ref, _Ptr > &__last1, _II __first2)
- template<typename _Tp, typename _Up = _Tp>
  constexpr _Tp **__exchange** (_Tp &__obj, _Up &&__new_val)
- template<typename _FIte, typename _Tp>
  constexpr void **__fill_a** (_FIte __first, _FIte __last, const _Tp &__value)
- template<typename _Ite, typename _Seq, typename _Cat, typename _Tp>
  constexpr void **__fill_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const _Tp &)
- template<typename _Ite, typename _Cont, typename _Tp>
  constexpr void **__fill_a1** (::__gnu_cxx::__normal_iterator< _Ite, _Cont > __first, ::__gnu_cxx::__normal_↩
  iterator< _Ite, _Cont > __last, const _Tp &__value)
- constexpr void **__fill_a1** (::_Bit_iterator, ::_Bit_iterator, const bool &)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void **__fill_a1** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _Up, typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_byte< _Up >::__value &&(__are_same< _Up, _Tp >::__value||__↩
  memcpyable_integer< _Tp >::__width), void >::__type **__fill_a1** (_Up *__first, _Up *__last, const _Tp &__x)
- template<typename _Tp, typename _VTp>
  void **__fill_a1** (const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _VTp &__value)
- constexpr void **__fill_bvector** (_Bit_type *__v, unsigned int __first, unsigned int __last, bool __x) noexcept
- constexpr void **__fill_bvector_n** (_Bit_type *, size_t, bool) noexcept
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value, std::input_iterator_tag)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value, std::output_iterator_tag)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value, std::random_access_iterator_tag)
- template<typename _Ite, typename _Seq, typename _Cat, typename _Size, typename _Tp>
  constexpr ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **__fill_n_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &__first, _Size __n, const _Tp &__value, std::input_iterator_tag)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **__fill_n_a1** (_OutputIterator __first, _Size __n, const _Tp &__value)
- template<typename _Iterator, typename _Predicate>
  constexpr _Iterator **__find_if** (_Iterator __first, _Iterator __last, _Predicate __pred)
- template<typename _Tp, typename... _Types>
  constexpr size_t **__find_uniq_type_in_pack** ()
- from_chars_result **__from_chars_bfloat16_t** (const char *__first, const char *__last, float &__value, chars_format __fmt=chars_format::general) noexcept
- from_chars_result **__from_chars_float16_t** (const char *__first, const char *__last, float &__value, chars_format __fmt=chars_format::general) noexcept
- template<typename _UInt1, typename _UInt2, typename _UComm = __conditional_t<(sizeof(_UInt2) > sizeof(_UInt1)), _UInt2, _UInt1>>
  constexpr __gen_canon_log_res< _UInt1 > **__gen_canon_log** (_UInt1 __val, _UInt2 __base)
- template<typename _IntType, typename _UniformRandomBitGenerator>
  pair< _IntType, _IntType > __gen_two_uniform_ints (_IntType __b0, _IntType __b1, _UniformRandomBit↩
  Generator &&__g)

- template<typename _RealT, size_t __d, typename _Urbg>
  _RealT **__generate_canonical_any** (_Urbg &__urng)
- template<typename _RealT, size_t __d, typename _Urbg>
  _RealT **__generate_canonical_pow2** (_Urbg &__urng)
- template<size_t __i, typename _Head, typename... _Tail>
  constexpr _Head & **__get_helper** (_Tuple_impl< __i, _Head, _Tail... > &__t) noexcept
- template<size_t __i, typename _Head, typename... _Tail>
  constexpr const _Head & **__get_helper** (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept
- template<size_t __i, typename... _Types>
  __enable_if_t<(__i >=sizeof...(_Types))> **__get_helper** (const tuple< _Types... > &)=delete
- void **__glibcxx_assert_fail** (const char *__file, int __line, const char *__function, const char *__condition) noexcept
- template<typename _Tp>
  size_t **__iconv_adaptor** (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **__inbuf, size_t *__inbytes, char **__outbuf, size_t *__outbytes)
- template<typename _CharT, typename _ValueT>
  int **__int_to_char** (_CharT *__bufend, _ValueT __v, const _CharT *__lit, ios_base::fmtflags __flags, bool __dec)
- template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>
  constexpr _Up && **__invfwd** (typename remove_reference< _Tp >::type &__t) noexcept
- template<typename _Callable, typename... _Args>
  constexpr __invoke_result< _Callable, _Args... >::type __invoke (_Callable &&__fn, _Args &&... __args) noexcept(__is_nothrow_invocable< _Callable, _Args... >::value)
- template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>
  constexpr _Res **__invoke_impl** (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)
- template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>
  constexpr _Res **__invoke_impl** (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)
- template<typename _Res, typename _MemPtr, typename _Tp>
  constexpr _Res **__invoke_impl** (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)
- template<typename _Res, typename _MemPtr, typename _Tp>
  constexpr _Res **__invoke_impl** (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)
- template<typename _Res, typename _Fn, typename... _Args>
  constexpr _Res **__invoke_impl** (__invoke_other, _Fn &&__f, _Args &&... __args)
- template<typename _Res, typename _Callable, typename... _Args>
  constexpr enable_if_t< is_invocable_r_v< _Res, _Callable, _Args... >, _Res > **__invoke_r** (_Callable &&__fn, _Args &&... __args) noexcept(is_nothrow_invocable_r_v< _Res, _Callable, _Args... >)
- constexpr bool **__is_constant_evaluated** () noexcept
- template<typename _RandomAccessIterator, typename _Compare, typename _Distance>
  constexpr bool **__is_heap** (_RandomAccessIterator __first, _Compare __comp, _Distance __n)
- template<typename _RandomAccessIterator, typename _Distance>
  constexpr bool **__is_heap** (_RandomAccessIterator __first, _Distance __n)
- template<typename _RandomAccessIterator>
  constexpr bool **__is_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr bool **__is_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Distance, typename _Compare>
  constexpr _Distance **__is_heap_until** (_RandomAccessIterator __first, _Distance __n, _Compare &__comp)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool **__is_permutation** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 _↩
  _first2, _BinaryPredicate __pred)
- template<typename _Rng>
  constexpr bool **__is_power_of_2_less_1** (_Rng __range)
- template<typename _CharT, typename _Traits>
  void **__istream_extract** (basic_istream< _CharT, _Traits > &, _CharT *, streamsize)

- void **__istream_extract** (istream &, char *, streamsize)
- template<typename _Tp1, typename _Ref, typename _Ptr, typename _Tp2>
  int **__lex_cmp_dit** (::_Deque_iterator< _Tp1, _Ref, _Ptr > __first1, ::_Deque_iterator< _Tp1, _Ref, _Ptr > _↩
  _last1, const _Tp2 *__first2, const _Tp2 *__last2)
- template<typename _II1, typename _II2>
  constexpr bool **__lexicographical_compare_aux** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1, typename _Ite2, typename _Seq2, typename _Cat2>
  constexpr bool **__lexicographical_compare_aux** (_II1 __first1, _II1 __last1, const ::__gnu_debug::_Safe_iterator<
  _Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::_Safe_iterator< _Ite2, _Seq2, _Cat2 > &__last2)
- template<typename _II1, typename _Iter2, typename _Seq2, typename _Cat2>
  constexpr bool **__lexicographical_compare_aux** (_II1, _II1, const ::__gnu_debug::_Safe_iterator< _Iter2, _↩
  Seq2, _Cat2 > &, const ::__gnu_debug::_Safe_iterator< _Iter2, _Seq2, _Cat2 > &)
- template<typename _Ite1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, _Cat1
  > &__first1, const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, _II2 __first2, _II2 __last2)
- template<typename _Ite1, typename _Seq1, typename _Cat1, typename _Ite2, typename _Seq2, typename _Cat2>
  constexpr bool **__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, ↩
  _Cat1 > &__first1, const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, const
  ::__gnu_debug::_Safe_iterator< _Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::_Safe_iterator< ↩
  _Ite2, _Seq2, _Cat2 > &__last2)
- template<typename _Iter1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _Cat1
  > &, const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _Cat1 > &, _II2, _II2)
- template<typename _Iter1, typename _Seq1, typename _Cat1, typename _Iter2, typename _Seq2, typename _Cat2>
  constexpr bool **__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _Cat1
  > &, const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator<
  _Iter2, _Seq2, _Cat2 > &, const ::__gnu_debug::_Safe_iterator< _Iter2, _Seq2, _Cat2 > &)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **__lexicographical_compare_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator<
  _Tp1, _Ref1, _Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::_Deque_iterator< _Tp2,
  _Ref2, _Ptr2 > __last2)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2>
  bool **__lexicographical_compare_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator<
  _Tp1, _Ref1, _Ptr1 > __last1, _Tp2 *__first2, _Tp2 *__last2)
- template<typename _II1, typename _II2>
  constexpr bool **__lexicographical_compare_aux1** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _Tp1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **__lexicographical_compare_aux1** (_Tp1 *__first1, _Tp1 *__last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2
  > __first2, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)
- template<typename _II1, typename _II2, typename _Compare>
  constexpr bool **__lexicographical_compare_impl** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _↩
  Compare __comp)
- template<typename _Tp>
  constexpr _Tp __lg (_Tp __n)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator **__lower_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val,
  _Compare __comp)
- template<typename... _BoundArgs, typename... _Args>
  constexpr auto **__make_bound_args** (_Args &&... __args)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **__make_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &_↩
  _comp)

- template<typename _Iterator, typename _ReturnType = __conditional_t<__move_if_noexcept_cond <typename iterator_traits<_↩ Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>>
  constexpr _ReturnType **__make_move_if_noexcept_iterator** (_Iterator __i)
- template<typename _Tp, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_↩ iterator<_Tp*>>>
  constexpr _ReturnType **__make_move_if_noexcept_iterator** (_Tp *__i)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > **__make_reverse_iterator** (_Iterator __i)
- template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename... _Args>
  __shared_ptr< _Tp, _Lp > **__make_shared** (_Args &&... __args)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator **__max_element** (_ForwardIterator __first, _ForwardIterator __last, _Compare __↩ comp)
- template<typename _Tp, typename _Up>
  constexpr int **__memcmp** (const _Tp *__first1, const _Up *__first2, size_t __num)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **__merge** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _↩ InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _Tp>
  constexpr auto **__min_cmp** (_Tp __x, _Tp __y)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator **__min_element** (_ForwardIterator __first, _ForwardIterator __last, _Compare __↩ comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > **__mismatch** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > **__mismatch** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)
- template<typename _Iterator>
  constexpr _Iterator **__miter_base** (_Iterator __it)
- template<typename _Ite, typename _Seq>
  constexpr decltype(std::__niter_base(std::declval< _Ite >())) **__niter_base** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, std::random_access_iterator_tag > &__it) noexcept(std::is_nothrow_copy_constructible< _Ite >::↩ value)
- void **__once_proxy** (void)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **__pop_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Random↩ AccessIterator __result, _Compare &__comp)
- template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>
  constexpr void **__push_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __topIndex, _Tp __value, _Compare &__comp)
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
  _Out_iter **__regex_replace** (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _↩ Rx_traits > &__e, const _Ch_type *__fmt, size_t __len, regex_constants::match_flag_type __flags)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
  constexpr _OutputIterator **__remove_copy_if** (_InputIterator __first, _InputIterator __last, _OutputIterator __↩ result, _Predicate __pred)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator **__remove_if** (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 **__search** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)

- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **__set_difference** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 _↩
  _first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **__set_intersection** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **__set_symmetric_difference** (_InputIterator1 __first1, _InputIterator1 __last1, _↩
  InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _Tp>
  _Tp * **__shared_ptr_deref** (_Tp *__p)
- constexpr long long **__size_to_integer** (double __n)
- constexpr long long **__size_to_integer** (float __n)
- constexpr int **__size_to_integer** (int __n)
- constexpr long **__size_to_integer** (long __n)
- constexpr long long **__size_to_integer** (long double __n)
- constexpr long long **__size_to_integer** (long long __n)
- constexpr unsigned **__size_to_integer** (unsigned __n)
- constexpr unsigned long **__size_to_integer** (unsigned long __n)
- constexpr unsigned long long **__size_to_integer** (unsigned long long __n)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **__sort_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__↩
  comp)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **__stable_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __↩
  comp)
- template<typename _Str>
  constexpr _Str **__str_concat** (typename _Str::value_type const *__lhs, typename _Str::size_type __lhs_len, typename _Str::value_type const *__rhs, typename _Str::size_type __rhs_len, typename _Str::allocator_type const &__a)
- constexpr size_t **__sv_check** (size_t __size, size_t __pos, const char *__s)
- constexpr size_t **__sv_limit** (size_t __size, size_t __pos, size_t __off) noexcept
- void **__terminate** () noexcept
- void **__throw_bad_alloc** (void)
- void **__throw_bad_array_new_length** (void)
- void **__throw_bad_cast** (void)
- void **__throw_bad_exception** (void)
- void **__throw_bad_function_call** ()
- void **__throw_bad_typeid** (void)
- void **__throw_bad_weak_ptr** ()
- void **__throw_domain_error** (const char *)
- void **__throw_future_error** (int)
- void **__throw_invalid_argument** (const char *)
- void **__throw_ios_failure** (const char *)
- void **__throw_ios_failure** (const char *, int)
- void **__throw_length_error** (const char *)
- void **__throw_logic_error** (const char *)
- void **__throw_out_of_range** (const char *)
- void **__throw_out_of_range_fmt** (const char *,...)
- void **__throw_overflow_error** (const char *)
- void **__throw_range_error** (const char *)
- void **__throw_runtime_error** (const char *)

- void **__throw_system_error** (int)
- void **__throw_underflow_error** (const char *)
- template<typename _Tp>
  constexpr to_chars_result **__to_chars_i** (char *__first, char *__last, _Tp __value, int __base=10)
- constexpr void **__to_wstring_numeric** (const char *__s, int __len, wchar_t *__wout)
- wstring **__to_wstring_numeric** (const string &__s)
- template<typename _Facet>
  const _Facet * **__try_use_facet** (const locale &__loc)
- template<typename _Cat, typename _Tp, typename _Up, typename _IndexSeq>
  constexpr _Cat **__tuple_cmp** (const _Tp &__t, const _Up &__u, _IndexSeq __indices)
- template<typename _Tp, typename _ForwardIterator>
  void **__uninitialized_construct_buf** (_Tp *__first, _Tp *__last, _ForwardIterator __seed)
- template<typename _ITp, typename _IRef, typename _IPtr, typename _OTp, typename _Tp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **__uninitialized_copy_a** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result, allocator< _Tp > &)
- template<typename _Iter, typename _OTp, typename _Tp>
  __enable_if_t< __is_random_access_iter< _Iter >::value, ::_Deque_iterator< _OTp, _OTp &, _OTp * > > ↩
  __uninitialized_copy_a (_Iter __first, _Iter __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result, allocator< _Tp > &)
- template<typename _ITp, typename _IRef, typename _IPtr, typename _OTp, typename _Tp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **__uninitialized_move_a** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result, allocator< _Tp > &)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator **__unique** (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↩
  binary_pred)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)
- template<typename _Tp>
  void **__valarray_copy** (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)

- template<typename _Tp, class _Dom>
  void **__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)
- template<typename _Tp, class _Dom>
  void **__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)
- template<typename _Tp, class _Dom>
  void **__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)
- template<typename _Tp, class _Dom>
  void **__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)
- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __a, _Tp *__restrict__ __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __a, const size_t *__restrict__ __i, _Tp *__restrict__ __b, size_t __n)
- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __a, size_t __n, _Tp *__restrict__ __b)
- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __a, size_t __n, _Tp *__restrict__ __b, const size_t *__restrict__ __i)
- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __a, size_t __n, size_t __s, _Tp *__restrict__ __b)
- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __src, size_t __n, const size_t *__restrict__ __i, _Tp *__restrict__ __dst, const size_t *__restrict__ __j)
- template<typename _Tp>
  void **__valarray_copy** (const _Tp *__restrict__ __src, size_t __n, size_t __s1, _Tp *__restrict__ __dst, size_t __s2)
- template<typename _Tp>
  void **__valarray_copy_construct** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **__valarray_copy_construct** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **__valarray_copy_construct** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **__valarray_copy_construct** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)
- template<typename _Tp>
  void **__valarray_copy_construct** (const _Tp *__b, const _Tp *__e, _Tp *__restrict__ __o)
- template<typename _Tp>
  void **__valarray_copy_construct** (const _Tp *__restrict__ __a, const size_t *__restrict__ __i, _Tp *__restrict__ __o, size_t __n)
- template<typename _Tp>
  void **__valarray_copy_construct** (const _Tp *__restrict__ __a, size_t __n, size_t __s, _Tp *__restrict__ __o)
- template<typename _Tp>
  void **__valarray_default_construct** (_Tp *__b, _Tp *__e)
- template<typename _Tp>
  void **__valarray_destroy_elements** (_Tp *__b, _Tp *__e)
- template<typename _Tp>
  void **__valarray_fill** (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **__valarray_fill** (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)
- template<typename _Tp>
  void **__valarray_fill** (_Array< _Tp > __a, size_t __n, const _Tp &__t)

- template<typename _Tp>
  void **__valarray_fill** (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)
- template<typename _Tp>
  void **__valarray_fill** (_Tp *__restrict__ __a, const size_t *__restrict__ __i, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **__valarray_fill** (_Tp *__restrict__ __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **__valarray_fill** (_Tp *__restrict__ __a, size_t __n, size_t __s, const _Tp &__t)
- template<typename _Tp>
  void **__valarray_fill_construct** (_Tp *__b, _Tp *__e, const _Tp __t)
- template<typename _Tp>
  _Tp * **__valarray_get_storage** (size_t)
- template<typename _Ta>
  _Ta::value_type **__valarray_max** (const _Ta &__a)
- template<typename _Ta>
  _Ta::value_type **__valarray_min** (const _Ta &__a)
- template<typename _Tp>
  void **__valarray_release_memory** (_Tp *__p, size_t __n)
- template<typename _Tp>
  _Tp **__valarray_sum** (const _Tp *__f, const _Tp *__l)
- template<template< typename > class _Trait, typename _Tp, typename _Up = _Tp>
  constexpr _Up **__value_or** (_Up __def=_Up()) noexcept
- bool **__verify_grouping** (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()
- template<typename _CharT, typename _OutIter>
  _OutIter **__write** (_OutIter __s, const _CharT *__ws, int __len)
- template<typename _CharT>
  ostreambuf_iterator< _CharT > **__write** (ostreambuf_iterator< _CharT > __s, const _CharT *__ws, int __len)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)

- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)

- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___divides** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___divides** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___divides** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)

- template<typename _Tp, class _Dom>
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___minus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___minus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___minus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)

- template<typename _Tp, class _Dom>
  void **_Array_augmented___modulus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___plus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___plus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)

- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___plus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)

- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, typename... _Args>
  constexpr void _Construct (_Tp *__p, _Args &&... __args)
- template<typename _T1>
  constexpr void **_Construct_novalue** (_T1 *__p)
- template<typename _ForwardIterator>
  constexpr void _Destroy (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _Tp>
  constexpr void _Destroy (_Tp *__pointer)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator _Destroy_n (_ForwardIterator __first, _Size __count)
- size_t **_Fnv_hash_bytes** (const void *__ptr, size_t __len, size_t __seed)
- size_t **_Hash_bytes** (const void *__ptr, size_t __len, size_t __seed)
- unsigned int **_Rb_tree_black_count** (const _Rb_tree_node_base *__node, const _Rb_tree_node_base *__root) throw ()
- _Rb_tree_node_base * **_Rb_tree_decrement** (_Rb_tree_node_base *__x) throw ()
- _Rb_tree_node_base * **_Rb_tree_increment** (_Rb_tree_node_base *__x) throw ()
- void **_Rb_tree_insert_and_rebalance** (const bool __insert_left, _Rb_tree_node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header) throw ()
- _Rb_tree_node_base * **_Rb_tree_rebalance_for_erase** (_Rb_tree_node_base *const __z, _Rb_tree_node_base &__header) throw ()
- template<class _Dom>
  _Expr< _UnClos< struct std::_Abs, _Expr, _Dom >, typename _Dom::value_type > **abs** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<signed_integral T, typename Abi>
  constexpr basic_vec< T, Abi > **abs** (const basic_vec< T, Abi > &__x)
- template<typename _Tp>
  _Tp abs (const complex< _Tp > &)
- template<typename _Tp>
  _Tp abs (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Abs, _ValArray, _Tp >, _Tp > **abs** (const valarray< _Tp > &__v)
- constexpr double **abs** (double __x)

- constexpr float **abs** (float __x)
- long **abs** (long __i)
- constexpr long double **abs** (long double __x)
- long long **abs** (long long __x)
- template<typename _InputIterator, typename _Tp>
  constexpr _Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init)
- template<typename _InputIterator, typename _Tp, typename _BinaryOperation>
  constexpr _Tp accumulate (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **acos** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Acos, _Expr, _Dom >, typename _Dom::value_type > **acos** (const _Expr< _↩
  Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  std::complex< _Tp > **acos** (const std::complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Acos, _ValArray, _Tp >, _Tp > **acos** (const valarray< _Tp > &__v)
- constexpr float **acos** (float __x)
- constexpr long double **acos** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **acosh** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **acosh** (const std::complex< _Tp > &)
- constexpr float acosh (float __x)
- constexpr long double **acosh** (long double __x)
- template<typename _Tp>
  constexpr _Tp * addressof (_Tp &__r) noexcept
- template<typename _Tp>
  const _Tp * **addressof** (const _Tp &&)=delete
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __↩
  result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator adjacent_difference (_InputIterator __first, _InputIterator __last, _OutputIterator __↩
  result, _BinaryOperation __binary_op)
- template<typename _FIter>
  constexpr _FIter **adjacent_find** (_FIter, _FIter)
- template<typename _FIter, typename _BinaryPredicate>
  constexpr _FIter **adjacent_find** (_FIter, _FIter, _BinaryPredicate)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate ↩
  __binary_pred)
- template<typename _InputIterator, typename _Distance>
  constexpr void advance (_InputIterator &__i, _Distance __n)
- template<typename _CharT, typename _Distance>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, void >::__type **advance** (istreambuf_iterator< _↩
  CharT > &__i, _Distance __n)
- void * align (size_t __align, size_t __size, void *&__ptr, size_t &__space) noexcept
- template<typename _IIter, typename _Predicate>
  constexpr bool **all_of** (_IIter, _IIter, _Predicate)

- template<typename _InputIterator, typename _Predicate>
  constexpr bool all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  constexpr bool **any_of** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type **arg** (_Tp __x)
- template<__simd_complex _Vp>
  constexpr rebind_t< __simd_complex_value_type< _Vp >, _Vp > **arg** (const _Vp &__x) noexcept
- template<typename _Tp>
  _Tp arg (const complex< _Tp > &)
- template<typename _Tp, typename... _Up>
  array (_Tp, _Up...) -> array< enable_if_t<(is_same_v< _Tp, _Up > &&...), _Tp >, 1+sizeof...(_Up)>
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **asin** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Asin, _Expr, _Dom >, typename _Dom::value_type > **asin** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  std::complex< _Tp > **asin** (const std::complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Asin, _ValArray, _Tp >, _Tp > **asin** (const valarray< _Tp > &__v)
- constexpr float **asin** (float __x)
- constexpr long double **asin** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **asinh** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **asinh** (const std::complex< _Tp > &)
- constexpr float **asinh** (float __x)
- constexpr long double **asinh** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float assoc_laguerref (unsigned int __n, unsigned int __m, float __x)
- long double assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float assoc_legendref (unsigned int __l, unsigned int __m, float __x)
- long double assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)
- template<size_t _Align, class _Tp>
  constexpr _Tp * assume_aligned (_Tp *__ptr) noexcept
- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > async (_Fn &&__fn, _Args &&... __args)
- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > async (launch __policy, _Fn &&__fn, _Args &&... __args)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **atan** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Atan, _Expr, _Dom >, typename _Dom::value_type > **atan** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  std::complex< _Tp > **atan** (const std::complex< _Tp > &)

- template<typename _Tp>
  _Expr< _UnClos< struct std::_Atan, _ValArray, _Tp >, _Tp > **atan** (const valarray< _Tp > &__v)
- constexpr float **atan** (float __x)
- constexpr long double **atan** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **atan2** (_Tp __y, _Up __x)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > **atan2** (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > **atan2** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::_Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > **atan2** (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > **atan2** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > **atan2** (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > **atan2** (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > **atan2** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > **atan2** (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr float **atan2** (float __y, float __x)
- constexpr long double **atan2** (long double __y, long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **atanh** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **atanh** (const std::complex< _Tp > &)
- constexpr float **atanh** (float __x)
- constexpr long double **atanh** (long double __x)
- template<typename _ITp>
  bool **atomic_compare_exchange_strong** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **atomic_compare_exchange_strong** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **atomic_compare_exchange_strong_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, _atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept

- template<typename _ITp>
  bool **atomic_compare_exchange_strong_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *↩ __i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  bool **atomic_compare_exchange_weak** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __atomic_↩ val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **atomic_compare_exchange_weak** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __↩ atomic_val_t< _ITp > __i2) noexcept
- template<typename _ITp>
  bool **atomic_compare_exchange_weak_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > *__i1, __↩ atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  bool **atomic_compare_exchange_weak_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > *↩ __i1, __atomic_val_t< _ITp > __i2, memory_order __m1, memory_order __m2) noexcept
- template<typename _ITp>
  _ITp **atomic_exchange** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_exchange** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_exchange_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_exchange_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_add** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_add** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_add_explicit** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_add_explicit** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_and** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_and** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_and_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_and_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __↩ i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_or** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_or** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_or_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept

- template<typename _ITp>
  _ITp **atomic_fetch_or_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_sub** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_sub** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_sub_explicit** (atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_sub_explicit** (volatile atomic< _ITp > *__a, __atomic_diff_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_xor** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_xor** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_xor_explicit** (__atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_fetch_xor_explicit** (volatile __atomic_base< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- void **atomic_flag_clear** (atomic_flag *__a) noexcept
- void **atomic_flag_clear** (volatile atomic_flag *__a) noexcept
- void **atomic_flag_clear_explicit** (atomic_flag *__a, memory_order __m) noexcept
- void **atomic_flag_clear_explicit** (volatile atomic_flag *__a, memory_order __m) noexcept
- bool **atomic_flag_test_and_set** (atomic_flag *__a) noexcept
- bool **atomic_flag_test_and_set** (volatile atomic_flag *__a) noexcept
- bool **atomic_flag_test_and_set_explicit** (atomic_flag *__a, memory_order __m) noexcept
- bool **atomic_flag_test_and_set_explicit** (volatile atomic_flag *__a, memory_order __m) noexcept
- template<typename _ITp>
  void **atomic_init** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **atomic_init** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  bool **atomic_is_lock_free** (const atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  bool **atomic_is_lock_free** (const volatile atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **atomic_load** (const atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **atomic_load** (const volatile atomic< _ITp > *__a) noexcept
- template<typename _ITp>
  _ITp **atomic_load_explicit** (const atomic< _ITp > *__a, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **atomic_load_explicit** (const volatile atomic< _ITp > *__a, memory_order __m) noexcept
- void **atomic_signal_fence** (memory_order __m) noexcept
- template<typename _ITp>
  void **atomic_store** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **atomic_store** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept

- template<typename _ITp>
  void **atomic_store_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  void **atomic_store_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- void **atomic_thread_fence** (memory_order __m) noexcept
- template<typename _Container>
  constexpr back_insert_iterator< _Container > back_inserter (_Container &__x)
- template<typename _ForwardIterator>
  basic_regex (_ForwardIterator, _ForwardIterator, regex_constants::syntax_option_type={}) -> basic_regex< typename iterator_traits< _ForwardIterator >::value_type >
- template<typename _InputIterator, typename _CharT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_CharT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  basic_string (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> basic_string< _CharT, char_traits< _CharT >, _Allocator >
- template<typename _CharT, typename _Traits, typename _Allocator = allocator<_CharT>, typename = _RequireAllocator<_Allocator>>
  basic_string (basic_string_view< _CharT, _Traits >, const _Allocator &=_Allocator()) -> basic_string< _CharT, _Traits, _Allocator >
- template<typename _CharT, typename _Traits, typename _Allocator = allocator<_CharT>, typename = _RequireAllocator<_Allocator>>
  basic_string (basic_string_view< _CharT, _Traits >, typename basic_string< _CharT, _Traits, _Allocator >::size_type, typename basic_string< _CharT, _Traits, _Allocator >::size_type, const _Allocator &=_Allocator()) -> basic_string< _CharT, _Traits, _Allocator >
- template<typename _Operation>
  void **basic_string< _CharT, _Traits, _Alloc >__resize_and_overwrite** (const size_type __n, _Operation __op)
- template<contiguous_iterator _It, sized_sentinel_for< _It > _End>
  basic_string_view (_It, _End) -> basic_string_view< iter_value_t< _It > >
- template<ranges::contiguous_range _Range>
  basic_string_view (_Range &&) -> basic_string_view< ranges::range_value_t< _Range > >
- template<typename _Container>
  constexpr auto begin (_Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.begin())
- template<typename _Tp, size_t _Nm>
  constexpr _Tp * begin (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto begin (const _Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.begin())
- template<typename _Tpa, typename _Tpb>
  __gnu_cxx::__promote_2< _Tpa, _Tpb >::__type beta (_Tpa __a, _Tpb __b)
- float betaf (float __a, float __b)
- long double betal (long double __a, long double __b)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr bool **binary_search** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type, typename _Compare>
  constexpr bool **binary_search** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _ForwardIterator, typename _Tp>
  constexpr bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr bool binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _Result, typename _Func, typename... _BoundArgs>
  constexpr _Bindres_helper< _Result, _Func, _BoundArgs... >::type bind (_Func &&__f, _BoundArgs &&... __args)

- template<typename _Func, typename... _BoundArgs>
  constexpr _Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type bind (_Func &&__f, _BoundArgs &&... __args)
- template<typename _Operation, typename _Tp>
  binder1st< _Operation > bind1st (const _Operation &__fn, const _Tp &__x)
- template<typename _Operation, typename _Tp>
  binder2nd< _Operation > bind2nd (const _Operation &__fn, const _Tp &__x)
- template<__simd_unsigned_integer _Vp>
  constexpr _Vp **bit_ceil** (const _Vp &__v)
- template<__simd_unsigned_integer _Vp>
  constexpr _Vp **bit_floor** (const _Vp &__v) noexcept
- template<__simd_unsigned_integer _Vp>
  constexpr rebind_t< make_signed_t< typename _Vp::value_type >, _Vp > **bit_width** (const _Vp &__v) noexcept
- ios_base & boolalpha (ios_base &__base)
- template<__simd_integral _Vp>
  constexpr _Vp **byteswap** (const _Vp &__v) noexcept
- template<typename _Callable, typename... _Args>
  void call_once (once_flag &__once, _Callable &&__f, _Args &&... __args)
- template<typename _IndexType, size_t... _Extents, typename... _RawSlices>
  constexpr auto **canonical_slices** (const extents< _IndexType, _Extents... > &__exts, _RawSlices... __raw_slices)
- template<typename _Container>
  constexpr auto cbegin (const _Container &__cont) noexcept(noexcept(std::begin(__cont))) -> decltype(std::begin(__cont))
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **cbrt** (_Tp __x)
- constexpr float **cbrt** (float __x)
- constexpr long double **cbrt** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **ceil** (_Tp __x)
- constexpr float **ceil** (float __x)
- constexpr long double **ceil** (long double __x)
- template<typename _Container>
  constexpr auto cend (const _Container &__cont) noexcept(noexcept(std::end(__cont))) -> decltype(std::end(__cont))
- template<typename _Tp>
  constexpr const _Tp & clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)
- template<typename _Tp, typename _Ap>
  constexpr basic_vec< _Tp, _Ap > **clamp** (const basic_vec< _Tp, _Ap > &__v, const basic_vec< _Tp, _Ap > &__lo, const basic_vec< _Tp, _Ap > &__hi)
- template<typename _Tp, typename _Up>
  constexpr bool **cmp_equal** (_Tp __t, _Up __u) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **cmp_greater** (_Tp __t, _Up __u) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **cmp_greater_equal** (_Tp __t, _Up __u) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **cmp_less** (_Tp __t, _Up __u) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **cmp_less_equal** (_Tp __t, _Up __u) noexcept

- template<typename _Tp, typename _Up>
  constexpr bool **cmp_not_equal** (_Tp __t, _Up __u) noexcept
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type comp_ellint_1 (_Tp __k)
- float comp_ellint_1f (float __k)
- long double comp_ellint_1l (long double __k)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type comp_ellint_2 (_Tp __k)
- float comp_ellint_2f (float __k)
- long double comp_ellint_2l (long double __k)
- template<typename _Tp, typename _Tpn>
  __gnu_cxx::__promote_2< _Tp, _Tpn >::__type comp_ellint_3 (_Tp __k, _Tpn __nu)
- float comp_ellint_3f (float __k, float __nu)
- long double comp_ellint_3l (long double __k, long double __nu)
- template<typename _Tp>
  constexpr std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **conj** (_Tp __x)
- template<__simd_complex _Vp>
  constexpr _Vp **conj** (const _Vp &__x) noexcept
- template<typename _Tp>
  constexpr complex< _Tp > conj (const complex< _Tp > &)
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > const_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, typename... _Args>
  constexpr _Tp * **construct_at** (_Tp *__location, _Args &&... __args) noexcept(noexcept(::new((void *) 0) _Tp(std::declval< _Args >()...)))
- template<typename _II, typename _OI>
  constexpr _OI copy (_II __first, _II __last, _OI __result)
- template<typename _IIter, typename _OIter>
  constexpr _OIter **copy** (_IIter, _IIter, _OIter)
- template<typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **copy** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)
- template<typename _BI1, typename _BI2>
  constexpr _BI2 copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _BIter1, typename _BIter2>
  constexpr _BIter2 **copy_backward** (_BIter1, _BIter1, _BIter2)
- template<typename _IIter, typename _OIter, typename _Predicate>
  constexpr _OIter **copy_if** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
  constexpr _OutputIterator copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _IIter, typename _Size, typename _OIter>
  constexpr _OIter **copy_n** (_IIter, _Size, _OIter)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  constexpr _OutputIterator copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **copysign** (_Tp __x, _Up __y)
- constexpr float **copysign** (float __x, float __y)
- constexpr long double **copysign** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **cos** (_Tp __x)

- template<class _Dom>
  _Expr< _UnClos< struct std::_Cos, _Expr, _Dom >, typename _Dom::value_type > **cos** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > cos (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Cos, _ValArray, _Tp >, _Tp > **cos** (const valarray< _Tp > &__v)
- constexpr float **cos** (float __x)
- constexpr long double **cos** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **cosh** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Cosh, _Expr, _Dom >, typename _Dom::value_type > **cosh** (const _Expr< _↩
  Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > cosh (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Cosh, _ValArray, _Tp >, _Tp > **cosh** (const valarray< _Tp > &__v)
- constexpr float **cosh** (float __x)
- constexpr long double **cosh** (long double __x)
- template<typename _IIter, typename _Tp = typename iterator_traits<_IIter>::value_type>
  constexpr iterator_traits< _IIter >::difference_type **count** (_IIter, _IIter, const _Tp &)
- template<typename _InputIterator, typename _Tp>
  constexpr iterator_traits< _InputIterator >::difference_type count (_InputIterator __first, _InputIterator __last, const _Tp &__value)
- template<typename _IIter, typename _Predicate>
  constexpr iterator_traits< _IIter >::difference_type **count_if** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator, typename _Predicate>
  constexpr iterator_traits< _InputIterator >::difference_type count_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<__simd_unsigned_integer _Vp>
  constexpr rebind_t< make_signed_t< typename _Vp::value_type >, _Vp > **countl_one** (const _Vp &__v) noexcept
- template<__simd_unsigned_integer _Vp>
  constexpr rebind_t< make_signed_t< typename _Vp::value_type >, _Vp > **countl_zero** (const _Vp &__v) noexcept
- template<__simd_unsigned_integer _Vp>
  constexpr rebind_t< make_signed_t< typename _Vp::value_type >, _Vp > **countr_one** (const _Vp &__v) noexcept
- template<__simd_unsigned_integer _Vp>
  constexpr rebind_t< make_signed_t< typename _Vp::value_type >, _Vp > **countr_zero** (const _Vp &__v) noexcept
- template<typename _Container>
  constexpr auto crbegin (const _Container &__cont) noexcept(noexcept(std::rbegin(__cont))) -> decltype(std::rbegin(↩
  __cont))
- template<typename _Container>
  constexpr auto crend (const _Container &__cont) noexcept(noexcept(std::rend(__cont))) -> decltype(std::rend(↩
  __cont))
- exception_ptr current_exception () noexcept
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float cyl_bessel_if (float __nu, float __x)

- long double cyl_bessel_il (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float cyl_bessel_jf (float __nu, float __x)
- long double cyl_bessel_jl (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float cyl_bessel_kf (float __nu, float __x)
- long double cyl_bessel_kl (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_neumann (_Tpnu __nu, _Tp __x)
- float cyl_neumannf (float __nu, float __x)
- long double cyl_neumannl (long double __nu, long double __x)
- template<typename _Container>
  constexpr auto data (_Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.data())
- template<typename _Tp, size_t _Nm>
  constexpr _Tp * data (_Tp(&__array)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto data (const _Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.data())
- ios_base & dec (ios_base &__base)
- template<typename _Tp>
  auto declval () noexcept -> decltype(__declval< _Tp >(0))
- ios_base & defaultfloat (ios_base &__base)
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  deque (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> deque< _ValT, _Allocator >
- template<typename _ForwardIterator>
  constexpr void **destroy** (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _Tp>
  constexpr void **destroy_at** (_Tp *__location)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator **destroy_n** (_ForwardIterator __first, _Size __count)
- template<typename _InputIterator>
  constexpr iterator_traits< _InputIterator >::difference_type distance (_InputIterator __first, _InputIterator __last)
- ldiv_t **div** (long __i, long __j) noexcept
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_1 (_Tp __k, _Tpp __phi)
- float ellint_1f (float __k, float __phi)
- long double ellint_1l (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_2 (_Tp __k, _Tpp __phi)
- float ellint_2f (float __k, float __phi)
- long double ellint_2l (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp>
  __gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float ellint_3f (float __k, float __nu, float __phi)
- long double ellint_3l (long double __k, long double __nu, long double __phi)
- template<typename _Container>
  constexpr auto empty (const _Container &__cont) noexcept(noexcept(__cont.empty())) -> decltype(__cont.↩
  empty())

- template<typename _Tp, size_t _Nm>
  constexpr bool empty (const _Tp(&)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto end (_Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())
- template<typename _Tp, size_t _Nm>
  constexpr _Tp * end (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto end (const _Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & endl (basic_ostream< _CharT, _Traits > &__os)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & ends (basic_ostream< _CharT, _Traits > &__os)
- template<typename _II1, typename _II2>
  constexpr bool equal (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _II1, typename _II2>
  constexpr bool equal (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _IIter2 __last2, _BinaryPredicate __binary↩
  _pred)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **equal** (_IIter1, _IIter1, _IIter2)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr pair< _FIter, _FIter > **equal_range** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type, typename _Compare>
  constexpr pair< _FIter, _FIter > **equal_range** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _ForwardIterator, typename _Tp>
  constexpr pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr pair< _ForwardIterator, _ForwardIterator > equal_range (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _Tp, size_t _Nm, typename _Up = _Tp>
  constexpr size_t **erase** (::inplace_vector< _Tp, _Nm > &__cont, const _Up &__value)
- template<typename _Tp, size_t _Nm, typename _Predicate>
  constexpr size_t **erase_if** (::inplace_vector< _Tp, _Nm > &__cont, _Predicate __pred)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **erf** (_Tp __x)
- constexpr float **erf** (float __x)
- constexpr long double **erf** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **erfc** (_Tp __x)
- constexpr float **erfc** (float __x)
- constexpr long double **erfc** (long double __x)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation>
  constexpr _OutputIterator exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init, _BinaryOperation __binary_op)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **exp** (_Tp __x)

- template<class _Dom>
  _Expr< _UnClos< struct std::_Exp, _Expr, _Dom >, typename _Dom::value_type > **exp** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > exp (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Exp, _ValArray, _Tp >, _Tp > **exp** (const valarray< _Tp > &__v)
- constexpr float **exp** (float __x)
- constexpr long double **exp** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **exp2** (_Tp __x)
- constexpr float **exp2** (float __x)
- constexpr long double **exp2** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type expint (_Tp __x)
- float expintf (float __x)
- long double expintl (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **expm1** (_Tp __x)
- constexpr float **expm1** (float __x)
- constexpr long double **expm1** (long double __x)
- template<typename... _Integrals>
  extents (_Integrals...) -> extents< size_t, __detail::__maybe_static_ext< _Integrals >... >
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **fabs** (_Tp __x)
- template<typename _Tp>
  _Tp fabs (const std::complex< _Tp > &__z)
- constexpr float **fabs** (float __x)
- constexpr long double **fabs** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **fdim** (_Tp __x, _Up __y)
- constexpr float **fdim** (float __x, float __y)
- constexpr long double **fdim** (long double __x, long double __y)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr void **fill** (_FIter, _FIter, const _Tp &)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _OI, typename _Size, typename _Tp>
  constexpr _OI fill_n (_OI __first, _Size __n, const _Tp &__value)
- template<typename _OIter, typename _Size, typename _Tp = typename iterator_traits<_OIter>::value_type>
  constexpr _OIter **fill_n** (_OIter, _Size, const _Tp &)
- template<typename _IIter, typename _Tp = typename iterator_traits<_IIter>::value_type>
  constexpr _IIter **find** (_IIter, _IIter, const _Tp &)
- template<typename _InputIterator, typename _Tp>
  constexpr _InputIterator find (_InputIterator __first, _InputIterator __last, const _Tp &__val)
- template<typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type **find** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter1 **find_end** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr _FIter1 **find_end** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)

- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter1 **find_first_of** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr _FIter1 **find_first_of** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _InputIterator, typename _ForwardIterator>
  constexpr _InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)
- template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _InputIterator find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)
- template<typename _IIter, typename _Predicate>
  constexpr _IIter **find_if** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator, typename _Predicate>
  constexpr _InputIterator find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  constexpr _IIter **find_if_not** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator, typename _Predicate>
  constexpr _InputIterator find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- ios_base & fixed (ios_base &__base)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **floor** (_Tp __x)
- constexpr float **floor** (float __x)
- constexpr long double **floor** (long double __x)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & flush (basic_ostream< _CharT, _Traits > &__os)
- template<typename _Tp, typename _Up, typename _Vp>
  constexpr __gnu_cxx::__promote_3< _Tp, _Up, _Vp >::__type **fma** (_Tp __x, _Up __y, _Vp __z)
- constexpr float **fma** (float __x, float __y, float __z)
- constexpr long double **fma** (long double __x, long double __y, long double __z)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **fmax** (_Tp __x, _Up __y)
- constexpr float **fmax** (float __x, float __y)
- constexpr long double **fmax** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **fmin** (_Tp __x, _Up __y)
- constexpr float **fmin** (float __x, float __y)
- constexpr long double **fmin** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **fmod** (_Tp __x, _Up __y)
- constexpr float **fmod** (float __x, float __y)
- constexpr long double **fmod** (long double __x, long double __y)
- template<typename _IIter, typename _Funct>
  constexpr _Funct **for_each** (_IIter, _IIter, _Funct)
- template<typename _InputIterator, typename _Function>
  constexpr _Function for_each (_InputIterator __first, _InputIterator __last, _Function __f)

- template<typename _InputIterator, typename _Size, typename _Function>
  constexpr _InputIterator for_each_n (_InputIterator __first, _Size __n, _Function __f)
- template<typename _Tp>
  constexpr _Tp && forward (typename std::remove_reference< _Tp >::type &&__t) noexcept
- template<typename _Tp>
  constexpr _Tp && forward (typename std::remove_reference< _Tp >::type &__t) noexcept
- template<typename... _Elements>
  constexpr tuple< _Elements &&... > forward_as_tuple (_Elements &&... __args) noexcept
- template<typename _Tp, typename _Up>
  constexpr __like_t< _Tp, _Up > forward_like (_Up &&__x) noexcept
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  forward_list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward_list< _ValT, _Allocator >
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, int >::__type **fpclassify** (_Tp __x)
- constexpr int **fpclassify** (double __x)
- constexpr int **fpclassify** (float __x)
- constexpr int **fpclassify** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **frexp** (_Tp __x, int *__exp)
- float **frexp** (float __x, int *__exp)
- long double **frexp** (long double __x, int *__exp)
- template<typename _Tp, enable_if_t< __or_< __is_signed_or_unsigned_integer< _Tp >, is_same< char, _Tp > >::value, int > = 0>
  constexpr from_chars_result from_chars (const char *__first, const char *__last, _Tp &__value, int __base=10)
- from_chars_result **from_chars** (const char *__first, const char *__last, double &__value, chars_format __fmt=chars_format::general) noexcept
- from_chars_result **from_chars** (const char *__first, const char *__last, float &__value, chars_format __fmt=chars_format::general) noexcept
- from_chars_result **from_chars** (const char *__first, const char *__last, long double &__value, chars_format __fmt=chars_format::general) noexcept
- template<typename _Container>
  constexpr front_insert_iterator< _Container > front_inserter (_Container &__x)
- template<typename _Fn, typename _Signature = __function_guide_t<_Fn, decltype(&_Fn::operator())>>
  function (_Fn) -> function< _Signature >
- template<typename _Res, typename... _ArgTypes>
  function (_Res(*)(_ArgTypes...)) -> function< _Res(_ArgTypes...)>
- template<typename _Fn>
  function_ref (_Fn *) -> function_ref< _Fn >
- template<auto __fn, typename _Fn>
  function_ref (constant_wrapper< __fn, _Fn >) -> function_ref< remove_pointer_t< _Fn > >
- template<auto __fn, typename _Fn, typename _Tp, typename _SignaturePtr = decltype(__polyfunc::__deduce_funcref<_Fn, _Tp&>())>
  function_ref (constant_wrapper< __fn, _Fn >, _Tp &&) -> function_ref< remove_pointer_t< _SignaturePtr > >
- const error_category & future_category () noexcept
- template<typename _FIter, typename _Generator>
  constexpr void **generate** (_FIter, _FIter, _Generator)
- template<typename _ForwardIterator, typename _Generator>
  constexpr void generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)
- template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>
  _RealType generate_canonical (_UniformRandomNumberGenerator &__g)
- template<typename _RealT, size_t __digits, typename _Urbg>
  _RealT generate_canonical (_Urbg &__urng)

- template<typename _OIter, typename _Size, typename _Generator>
  constexpr _OIter **generate_n** (_OIter, _Size, _Generator)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  constexpr _OutputIterator generate_n (_OutputIterator __first, _Size __n, _Generator __gen)
- const error_category & generic_category () noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr _Tp && **get** (array< _Tp, _Nm > &&__arr) noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr _Tp & **get** (array< _Tp, _Nm > &__arr) noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr const _Tp && **get** (const array< _Tp, _Nm > &&__arr) noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr const _Tp & **get** (const array< _Tp, _Nm > &__arr) noexcept
- template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>
  constexpr auto **get** (const subrange< _It, _Sent, _Kind > &__r)
- template<size_t __i, typename... _Elements>
  constexpr const __tuple_element_t< __i, tuple< _Elements... > > && get (const tuple< _Elements... > &&__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr const __tuple_element_t< __i, tuple< _Elements... > > & get (const tuple< _Elements... > &__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr __tuple_element_t< __i, tuple< _Elements... > > && get (tuple< _Elements... > &&__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr __tuple_element_t< __i, tuple< _Elements... > > & get (tuple< _Elements... > &__t) noexcept
- Catalogs & **get_catalogs** ()
- template<typename _Del, typename _Tp, _Lock_policy _Lp>
  _Del * **get_deleter** (const __shared_ptr< _Tp, _Lp > &__p) noexcept
- template<typename _Del, typename _Tp>
  _Del * get_deleter (const shared_ptr< _Tp > &__p) noexcept
- template<typename _MoneyT>
  _Get_money< _MoneyT > get_money (_MoneyT &__mon, bool __intl=false)
- new_handler get_new_handler () noexcept
- template<typename _Tp>
  pair< _Tp *, ptrdiff_t > get_temporary_buffer (ptrdiff_t __len) noexcept
- terminate_handler get_terminate () noexcept
- template<typename _CharT>
  _Get_time< _CharT > get_time (std::tm *__tmb, const _CharT *__fmt)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)

- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)
- template<> basic_istream< char > & **getline** (basic_istream< char > &__in, basic_string< char > &__str, char __delim)
- template<> basic_istream< wchar_t > & **getline** (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_t __delim)
- template<typename _Facet>
  bool has_facet (const locale &__loc)
- template bool **has_facet**< **codecvt**< **char, char, mbstate_t** > > (const locale &)
- template bool **has_facet**< **codecvt**< **wchar_t, char, mbstate_t** > > (const locale &)
- template<__simd_unsigned_integer _Vp>
  constexpr _Vp::mask_type **has_single_bit** (const _Vp &__v) noexcept
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type hermite (unsigned int __n, _Tp __x)
- float hermitef (unsigned int __n, float __x)
- long double hermitel (unsigned int __n, long double __x)
- ios_base & hex (ios_base &__base)
- ios_base & hexfloat (ios_base &__base)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **hypot** (_Tp __x, _Up __y)
- constexpr float **hypot** (float __x, float __y)
- constexpr long double **hypot** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, int >::__type **ilogb** (_Tp __x)
- constexpr int **ilogb** (float __x)
- constexpr int **ilogb** (long double __x)
- constexpr _RealSimd **imag** () const noexcept
- template<typename _Tp>
  constexpr __gnu_cxx::__promote< _Tp >::__type **imag** (_Tp)
- template<typename _Tp>
  constexpr _Tp **imag** (const complex< _Tp > &__z)
- template<typename _Res, typename _Tp>
  constexpr bool **in_range** (_Tp __t) noexcept
- template<typename _IIter1, typename _IIter2>
  constexpr bool **includes** (_IIter1, _IIter1, _IIter2, _IIter2)
- template<typename _IIter1, typename _IIter2, typename _Compare>
  constexpr bool **includes** (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _InputIterator1, typename _InputIterator2, typename _Compare>
  constexpr bool includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _Compare __comp)
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _Tp>
  constexpr _OutputIterator inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _Tp __init)

- template<typename _InputIterator1, typename _InputIterator2, typename _Tp>
  constexpr _Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)
- template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _Binary↩ Operation2>
  constexpr _Tp inner_product (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)
- template<typename _Pointer = void, typename _Smart, typename... _Args>
  auto inout_ptr (_Smart &__s, _Args &&... __args)
- template<typename _BidirectionalIterator>
  constexpr void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr void inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)
- template<typename _BIter>
  constexpr void **inplace_merge** (_BIter, _BIter, _BIter)
- template<typename _BIter, typename _Compare>
  constexpr void **inplace_merge** (_BIter, _BIter, _BIter, _Compare)
- template<typename _Container>
  constexpr insert_iterator< _Container > inserter (_Container &__x, std::__detail::__range_iter_t< _Container > __i)
- ios_base & internal (ios_base &__base)
- const error_category & **iostream_category** () noexcept
- template<typename _ForwardIterator, typename _Tp>
  constexpr void iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)
- constexpr bool **is_eq** (partial_ordering __cmp) noexcept
- constexpr bool **is_gt** (partial_ordering __cmp) noexcept
- constexpr bool **is_gteq** (partial_ordering __cmp) noexcept
- template<typename _RAIter>
  constexpr bool **is_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr bool **is_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr bool is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RAIter>
  constexpr _RAIter **is_heap_until** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr _RAIter **is_heap_until** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr _RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __↩ last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr _RandomAccessIterator is_heap_until (_RandomAccessIterator __first, _RandomAccessIterator __↩ last, _Compare __comp)
- constexpr bool **is_lt** (partial_ordering __cmp) noexcept
- constexpr bool **is_lteq** (partial_ordering __cmp) noexcept
- constexpr bool **is_neq** (partial_ordering __cmp) noexcept
- template<typename _IIter, typename _Predicate>
  constexpr bool **is_partitioned** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)

- template<typename _FIter1, typename _FIter2>
  constexpr bool **is_permutation** (_FIter1, _FIter1, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr bool **is_permutation** (_FIter1, _FIter1, _FIter2, _BinaryPredicate)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)
- template<typename _FIter>
  constexpr bool **is_sorted** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr bool **is_sorted** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator>
  constexpr bool is_sorted (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr bool is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _FIter>
  constexpr _FIter **is_sorted_until** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr _FIter **is_sorted_until** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<size_t _Align, class _Tp>
  bool is_sufficiently_aligned (_Tp *__ptr)
- template<typename _CharT>
  bool isalnum (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool isalpha (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool isblank (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool iscntrl (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool isdigit (_CharT __c, const locale &__loc)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **isfinite** (_Tp)
- constexpr bool **isfinite** (double __x)
- constexpr bool **isfinite** (float __x)
- constexpr bool **isfinite** (long double __x)
- template<typename _CharT>
  bool isgraph (_CharT __c, const locale &__loc)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value && __is_arithmetic< _Up >::__value), bool >::__type **isgreater** (_Tp __x, _Up __y)

- constexpr bool **isgreater** (double __x, double __y)
- constexpr bool **isgreater** (float __x, float __y)
- constexpr bool **isgreater** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **isgreaterequal** (_Tp __x, _Up __y)
- constexpr bool **isgreaterequal** (double __x, double __y)
- constexpr bool **isgreaterequal** (float __x, float __y)
- constexpr bool **isgreaterequal** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **isinf** (_Tp)
- constexpr bool **isinf** (double __x)
- constexpr bool **isinf** (float __x)
- constexpr bool **isinf** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **isless** (_Tp __x, _Up __y)
- constexpr bool **isless** (double __x, double __y)
- constexpr bool **isless** (float __x, float __y)
- constexpr bool **isless** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **islessequal** (_Tp __x, _Up __y)
- constexpr bool **islessequal** (double __x, double __y)
- constexpr bool **islessequal** (float __x, float __y)
- constexpr bool **islessequal** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **islessgreater** (_Tp __x, _Up __y)
- constexpr bool **islessgreater** (double __x, double __y)
- constexpr bool **islessgreater** (float __x, float __y)
- constexpr bool **islessgreater** (long double __x, long double __y)
- template<typename _CharT>
  bool islower (_CharT __c, const locale &__loc)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **isnan** (_Tp)
- constexpr bool **isnan** (double __x)
- constexpr bool **isnan** (float __x)
- constexpr bool **isnan** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **isnormal** (_Tp __x)
- constexpr bool **isnormal** (double __x)
- constexpr bool **isnormal** (float __x)
- constexpr bool **isnormal** (long double __x)
- template<typename _CharT>
  bool isprint (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool ispunct (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool isspace (_CharT __c, const locale &__loc)

- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **isunordered** (_Tp __x, _Up __y)
- constexpr bool **isunordered** (double __x, double __y)
- constexpr bool **isunordered** (float __x, float __y)
- constexpr bool **isunordered** (long double __x, long double __y)
- template<typename _CharT>
  bool isupper (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool isxdigit (_CharT __c, const locale &__loc)
- template<typename _FIter1, typename _FIter2>
  constexpr void **iter_swap** (_FIter1, _FIter2)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr void iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)
- template<typename _Tp>
  _Tp kill_dependency (_Tp __y) noexcept
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type laguerre (unsigned int __n, _Tp __x)
- float laguerref (unsigned int __n, float __x)
- long double laguerrel (unsigned int __n, long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **ldexp** (_Tp __x, int __exp)
- constexpr float **ldexp** (float __x, int __exp)
- constexpr long double **ldexp** (long double __x, int __exp)
- ios_base & left (ios_base &__base)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type legendre (unsigned int __l, _Tp __x)
- float legendref (unsigned int __l, float __x)
- long double legendrel (unsigned int __l, long double __x)
- template<typename _II1, typename _II2>
  constexpr bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1, typename _II2, typename _Compare>
  constexpr bool lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare __comp)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **lexicographical_compare** (_IIter1, _IIter1, _IIter2, _IIter2)
- template<typename _IIter1, typename _IIter2, typename _Compare>
  constexpr bool **lexicographical_compare** (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)
- template<typename _InputIter1, typename _InputIter2>
  constexpr auto **lexicographical_compare_three_way** (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 __↩
  first2, _InputIter2 __last2)
- template<typename _InputIter1, typename _InputIter2, typename _Comp>
  constexpr auto lexicographical_compare_three_way (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 __first2, _InputIter2 __last2, _Comp __comp) -> decltype(__comp(*__first1, *__first2))
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **lgamma** (_Tp __x)
- constexpr float **lgamma** (float __x)
- constexpr long double **lgamma** (long double __x)
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> list< _ValT, _Allocator >
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, longlong >::__type **llrint** (_Tp __x)

- constexpr long long **llrint** (float __x)
- constexpr long long **llrint** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, longlong >::__type **llround** (_Tp __x)
- constexpr long long **llround** (float __x)
- constexpr long long **llround** (long double __x)
- template<typename _L1, typename _L2, typename... _L3>
  void lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **log** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Log, _Expr, _Dom >, typename _Dom::value_type > **log** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > log (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Log, _ValArray, _Tp >, _Tp > **log** (const valarray< _Tp > &__v)
- constexpr float **log** (float __x)
- constexpr long double **log** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **log10** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Log10, _Expr, _Dom >, typename _Dom::value_type > **log10** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > log10 (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Log10, _ValArray, _Tp >, _Tp > **log10** (const valarray< _Tp > &__v)
- constexpr float **log10** (float __x)
- constexpr long double **log10** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **log1p** (_Tp __x)
- constexpr float **log1p** (float __x)
- constexpr long double **log1p** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **log2** (_Tp __x)
- constexpr float **log2** (float __x)
- constexpr long double **log2** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **logb** (_Tp __x)
- constexpr float **logb** (float __x)
- constexpr long double **logb** (long double __x)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr _FIter **lower_bound** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type, typename _Compare>
  constexpr _FIter **lower_bound** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, ↩
  _Compare __comp)

- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, long >::__type **lrint** (_Tp __x)
- constexpr long **lrint** (float __x)
- constexpr long **lrint** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, long >::__type **lround** (_Tp __x)
- constexpr long **lround** (float __x)
- constexpr long **lround** (long double __x)
- template<input_iterator _It>
  constexpr const_iterator< _It > **make_const_iterator** (_It __it) noexcept(is_nothrow_convertible_v< _It, const_iterator< _It > >)
- template<semiregular _Sent>
  constexpr const_sentinel< _Sent > **make_const_sentinel** (_Sent __s) noexcept(is_nothrow_convertible_v< _Sent, const_sentinel< _Sent > >)
- error_code make_error_code (errc __e) noexcept
- error_code make_error_code (future_errc __errc) noexcept
- error_code **make_error_code** (io_errc __e) noexcept
- error_condition make_error_condition (errc __e) noexcept
- error_condition make_error_condition (future_errc __errc) noexcept
- error_condition **make_error_condition** (io_errc __e) noexcept
- template<typename _Ex>
  constexpr exception_ptr make_exception_ptr (_Ex __ex) noexcept
- template<typename _RAIter>
  constexpr void **make_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **make_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void make_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _Iterator>
  constexpr move_iterator< _Iterator > **make_move_iterator** (_Iterator __i)
- template<typename _Tp, typename _Alloc, typename... _Args>
  constexpr _Tp **make_obj_using_allocator** (const _Alloc &__a, _Args &&... __args)
- template<typename _T1, typename _T2>
  constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > make_pair (_T1 &&__x, _T2 &&__y)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > make_reverse_iterator (_Iterator __i)
- template<typename... _Elements>
  constexpr tuple< typename __decay_and_strip< _Elements >::__type... > make_tuple (_Elements &&... __args)
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  map (_InputIterator, _InputIterator, _Allocator) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  map (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> map< _Key, _Tp, less< _Key >, _Allocator >

- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  map (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> map< _Key, _Tp, _Compare, _Allocator >
- template<typename _Tp>
  constexpr const _Tp & max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & max (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp, typename _Ap>
  constexpr basic_vec< _Tp, _Ap > **max** (const basic_vec< _Tp, _Ap > &__a, const basic_vec< _Tp, _Ap > &__b) noexcept
- template<typename _Tp>
  constexpr _Tp **max** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr _Tp **max** (initializer_list< _Tp >, _Compare)
- template<typename _FIter>
  constexpr _FIter **max_element** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr _FIter **max_element** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _CArray>
  mdspan (_CArray &) -> mdspan< remove_all_extents_t< _CArray >, extents< size_t, extent_v< _CArray, 0 > > >
- template<typename _ElementType, typename... _Integrals>
  mdspan (_ElementType *, _Integrals...) -> mdspan< _ElementType, extents< size_t, __detail::__maybe_static_ext< _Integrals >... > >
- template<typename _ElementType, typename _MappingType>
  mdspan (_ElementType *, const _MappingType &) -> mdspan< _ElementType, typename _MappingType::extents_type, typename _MappingType::layout_type >
- template<typename _ElementType, typename _OIndexType, size_t _Nm>
  mdspan (_ElementType *, const array< _OIndexType, _Nm > &) -> mdspan< _ElementType, dextents< size_t, _Nm > >
- template<typename _ElementType, typename _IndexType, size_t... _ExtentsPack>
  mdspan (_ElementType *, const extents< _IndexType, _ExtentsPack... > &) -> mdspan< _ElementType, extents< _IndexType, _ExtentsPack... > >
- template<typename _ElementType, typename _OIndexType, size_t _Nm>
  mdspan (_ElementType *, span< _OIndexType, _Nm >) -> mdspan< _ElementType, dextents< size_t, _Nm > >
- template<typename _Pointer>
  mdspan (_Pointer &&) -> mdspan< remove_pointer_t< remove_reference_t< _Pointer > >, extents< size_t > >
- template<typename _MappingType, typename _AccessorType>
  mdspan (typename _AccessorType::data_handle_type, const _MappingType &, const _AccessorType &) -> mdspan< typename _AccessorType::element_type, typename _MappingType::extents_type, typename _MappingType::layout_type, _AccessorType >
- template<typename _Tp, typename _Class>
  constexpr _Mem_fn< _Tp _Class::* > mem_fn (_Tp _Class::*__pm) noexcept
- template<typename _Ret, typename _Tp>
  const_mem_fun_t< _Ret, _Tp > **mem_fun** (_Ret(_Tp::*__f)() const)

- template<typename _Ret, typename _Tp>
  mem_fun_t< _Ret, _Tp > **mem_fun** (_Ret(_Tp::*__f)())
- template<typename _Ret, typename _Tp, typename _Arg>
  const_mem_fun1_t< _Ret, _Tp, _Arg > **mem_fun** (_Ret(_Tp::*__f)(_Arg) const)
- template<typename _Ret, typename _Tp, typename _Arg>
  mem_fun1_t< _Ret, _Tp, _Arg > **mem_fun** (_Ret(_Tp::*__f)(_Arg))
- template<typename _Ret, typename _Tp>
  const_mem_fun_ref_t< _Ret, _Tp > **mem_fun_ref** (_Ret(_Tp::*__f)() const)
- template<typename _Ret, typename _Tp>
  mem_fun_ref_t< _Ret, _Tp > **mem_fun_ref** (_Ret(_Tp::*__f)())
- template<typename _Ret, typename _Tp, typename _Arg>
  const_mem_fun1_ref_t< _Ret, _Tp, _Arg > **mem_fun_ref** (_Ret(_Tp::*__f)(_Arg) const)
- template<typename _Ret, typename _Tp, typename _Arg>
  mem_fun1_ref_t< _Ret, _Tp, _Arg > **mem_fun_ref** (_Ret(_Tp::*__f)(_Arg))
- void * **memchr** (void *__s, int __c, size_t __n)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _↩
  InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _↩
  InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _Tp>
  constexpr const _Tp & min (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & min (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp, typename _Ap>
  constexpr basic_vec< _Tp, _Ap > **min** (const basic_vec< _Tp, _Ap > &__a, const basic_vec< _Tp, _Ap > &__b) noexcept
- template<typename _Tp>
  constexpr _Tp **min** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr _Tp **min** (initializer_list< _Tp >, _Compare)
- template<typename _FIter>
  constexpr _FIter **min_element** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr _FIter **min_element** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _Tp>
  constexpr pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr pair< const _Tp &, const _Tp & > minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp, typename _Ap>
  constexpr pair< basic_vec< _Tp, _Ap >, basic_vec< _Tp, _Ap > > **minmax** (const basic_vec< _Tp, _Ap > &__a, const basic_vec< _Tp, _Ap > &__b) noexcept
- template<typename _Tp>
  constexpr pair< _Tp, _Tp > **minmax** (initializer_list< _Tp >)

- template<typename _Tp, typename _Compare>
  constexpr pair< _Tp, _Tp > **minmax** (initializer_list< _Tp >, _Compare)
- template<typename _FIter>
  constexpr pair< _FIter, _FIter > **minmax_element** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr pair< _FIter, _FIter > **minmax_element** (_FIter, _FIter, _Compare)
- template<typename _ForwardIterator>
  constexpr pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _Forward↩
  Iterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr pair< _ForwardIterator, _ForwardIterator > minmax_element (_ForwardIterator __first, _Forward↩
  Iterator __last, _Compare __comp)
- template<typename _IIter1, typename _IIter2>
  constexpr pair< _IIter1, _IIter2 > **mismatch** (_IIter1, _IIter1, _IIter2)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr pair< _IIter1, _IIter2 > **mismatch** (_IIter1, _IIter1, _IIter2, _BinaryPredicate)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, ↩
  _InputIterator2 __first2)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, ↩
  _InputIterator2 __first2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, ↩
  _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > mismatch (_InputIterator1 __first1, _InputIterator1 __last1, ↩
  _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)
- float **modf** (float __x, float *__iptr)
- long double **modf** (long double __x, long double *__iptr)
- template<typename _II, typename _OI>
  constexpr _OI move (_II __first, _II __last, _OI __result)
- template<typename _Tp>
  constexpr std::remove_reference< _Tp >::type && move (_Tp &&__t) noexcept
- template<typename _BI1, typename _BI2>
  constexpr _BI2 move_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _Tp>
  constexpr __conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > move_if_noexcept
  (_Tp &__x) noexcept
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩
  Allocator<_Allocator>>
  multimap (_InputIterator, _InputIterator, _Allocator) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t<
  _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter↩
  _to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename
  = _RequireAllocator<_Allocator>>
  multimap (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< __iter↩
  _key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> multimap< _Key, _Tp, less< _Key >, _Allocator
  >

- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multimap (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< _Key, _Tp, _Compare, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  multiset (_InputIterator, _InputIterator, _Allocator) -> multiset< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _↩ Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multiset (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  multiset (initializer_list< _Key >, _Allocator) -> multiset< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNot↩ Allocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multiset (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< _Key, _↩ Compare, _Allocator >
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **nearbyint** (_Tp __x)
- constexpr float **nearbyint** (float __x)
- constexpr long double **nearbyint** (long double __x)
- template<typename _InputIterator>
  constexpr _InputIterator **next** (_InputIterator __x, typename iterator_traits< _InputIterator >::difference_type _↩ _n=1)
- template<typename _BidirectionalIterator>
  constexpr bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _BIter>
  constexpr bool **next_permutation** (_BIter, _BIter)
- template<typename _BIter, typename _Compare>
  constexpr bool **next_permutation** (_BIter, _BIter, _Compare)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **nextafter** (_Tp __x, _Up __y)
- constexpr float **nextafter** (float __x, float __y)
- constexpr long double **nextafter** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **nexttoward** (_Tp __x, long double __y)
- constexpr float **nexttoward** (float __x, long double __y)
- constexpr long double **nexttoward** (long double __x, long double __y)
- ios_base & noboolalpha (ios_base &__base)
- template<typename _IIter, typename _Predicate>
  constexpr bool **none_of** (_IIter, _IIter, _Predicate)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _Tp>
  constexpr __gnu_cxx::__promote< _Tp >::__type **norm** (_Tp __x)
- template<__simd_complex _Vp>
  constexpr rebind_t< __simd_complex_value_type< _Vp >, _Vp > **norm** (const _Vp &__x) noexcept

- template<typename _Tp>
  _Tp constexpr norm (const complex< _Tp > &)
- template<typename _Tp>
  constexpr _Tp norm (const complex< _Tp > &__z)
- ios_base & noshowbase (ios_base &__base)
- ios_base & noshowpoint (ios_base &__base)
- ios_base & noshowpos (ios_base &__base)
- ios_base & noskipws (ios_base &__base)
- template<typename _Predicate>
  constexpr unary_negate< _Predicate > not1 (const _Predicate &__pred)
- template<typename _Predicate>
  constexpr binary_negate< _Predicate > not2 (const _Predicate &__pred)
- void **notify_all_at_thread_exit** (condition_variable &, unique_lock< mutex >)
- ios_base & nounitbuf (ios_base &__base)
- ios_base & nouppercase (ios_base &__base)
- template<typename _RAIter>
  constexpr void **nth_element** (_RAIter, _RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **nth_element** (_RAIter, _RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccess↩
  Iterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccess↩
  Iterator __last, _Compare __comp)
- ios_base & oct (ios_base &__base)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, type-
  name __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **operator!=** (const _↩
  Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, type-
  name __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **operator!=** (const _↩
  Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↩
  __not_equal_to, typename _Dom1::value_type >::result_type > **operator!=** (const _Expr< _Dom1, typename
  _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _StateT>
  bool **operator!=** (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare,
  _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc >
  &__y)
- template<typename _Tp, typename _Seq>
  bool operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() != __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator!=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &↩
  __y)

- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **operator!=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **operator!=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **operator%** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **operator%** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__modulus, typename _Dom1::value_type >::result_type > **operator%** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **operator%** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **operator%** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr _Ios_Fmtflags **operator&** (_Ios_Fmtflags __a, _Ios_Fmtflags __b) noexcept
- constexpr _Ios_Iostate **operator&** (_Ios_Iostate __a, _Ios_Iostate __b) noexcept
- constexpr _Ios_Openmode **operator&** (_Ios_Openmode __a, _Ios_Openmode __b) noexcept
- constexpr chars_format **operator&** (chars_format __lhs, chars_format __rhs) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **operator&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **operator&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__bitwise_and, typename _Dom1::value_type >::result_type > **operator&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **operator&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **operator&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr launch **operator&** (launch __x, launch __y) noexcept
- constexpr memory_order **operator&** (memory_order __m, __memory_order_modifier __mod) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__logical_and, typename _Dom1::value_type >::result_type > **operator&&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **operator&&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **operator&&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr const _Ios_Fmtflags & **operator&=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b) noexcept
- constexpr const _Ios_Iostate & **operator&=** (_Ios_Iostate &__a, _Ios_Iostate __b) noexcept
- constexpr const _Ios_Openmode & **operator&=** (_Ios_Openmode &__a, _Ios_Openmode __b) noexcept
- constexpr chars_format & **operator&=** (chars_format &__lhs, chars_format __rhs) noexcept
- constexpr launch & **operator&=** (launch &__x, launch __y) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **operator*** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **operator*** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__multiplies, typename _Dom1::value_type >::result_type > **operator*** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **operator*** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **operator*** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)

- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (const _CharT *__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _fun< struct std::__plus, typename _Dom::value_type >::result_type > **operator+** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _fun< struct std::__plus, typename _Dom::value_type >::result_type > **operator+** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__plus, typename _Dom1::value_type >::result_type > **operator+** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **operator+** (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _Tp>
  constexpr complex< _Tp > operator+ (const complex< _Tp > &__x)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _fun< struct std::__plus, typename _Dom::value_type >::result_type > **operator+** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩
  _fun< struct std::__plus, typename _Dom::value_type >::result_type > **operator+** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Iterator>
  requires requires { { __x.base() + __n } -> same_as<_Iterator>; }
  constexpr move_iterator< _Iterator > **operator+** (typename move_iterator< _Iterator >::difference_type __n, const move_iterator< _Iterator > &__x)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > **operator+** (typename reverse_iterator< _Iterator >::difference_type ↩
  __n, const reverse_iterator< _Iterator > &__x)
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _↩
  _fun< struct std::__minus, typename _Dom::value_type >::result_type > **operator-** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _↩
  _fun< struct std::__minus, typename _Dom::value_type >::result_type > **operator-** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__minus, typename _Dom1::value_type >::result_type > **operator-** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _Tp>
  constexpr complex< _Tp > operator- (const complex< _Tp > &__x)
- template<typename _IteratorL, typename _IteratorR>
  constexpr auto **operator-** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) -> decltype(__x.base() - __y.base())
- template<typename _IteratorL, typename _IteratorR>
  constexpr auto **operator-** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) -> decltype(__y.base() - __x.base())
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__minus, typename _Dom::value_type >::result_type > **operator-** (const typename _Dom::↩
  value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩
  _fun< struct std::__minus, typename _Dom::value_type >::result_type > **operator-** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename ↩
  __fun< struct std::__divides, typename _Dom::value_type >::result_type > **operator/** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__divides, typename _Dom::value_type >::result_type > **operator/** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__↩
  divides, typename _Dom1::value_type >::result_type > **operator/** (const _Expr< _Dom1, typename _Dom1::↩
  value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename

__fun< struct std::__divides, typename _Dom::value_type >::result_type > **operator/** (const typename _Dom::↩ value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩ _fun< struct std::__divides, typename _Dom::value_type >::result_type > **operator/** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _↩ _fun< struct std::__less, typename _Dom::value_type >::result_type > **operator**< (const _Expr< _Dom, type- name _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _↩ _fun< struct std::__less, typename _Dom::value_type >::result_type > **operator**< (const _Expr< _Dom, type- name _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__less, typename _Dom1::value_type >::result_type > **operator**< (const _Expr< _Dom1, typename _Dom1::value_↩ type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator**< (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() > __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator**< (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &↩ __y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _↩ _fun< struct std::__less, typename _Dom::value_type >::result_type > **operator**< (const typename _Dom::↩ value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩ _fun< struct std::__less, typename _Dom::value_type >::result_type > **operator**< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp, typename _Dp>
  constexpr bool operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const error_code &__e)

- template<typename _Ostream, typename _Tp>
  __rvalue_stream_insertion_t< _Ostream, _Tp > operator<< (_Ostream &&__os, const _Tp &__x)
- template<typename _CharT, typename _Traits, typename _MoneyT>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Put_money< _MoneyT > __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Put_time< _CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Resetiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setbase __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setfill< _CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setprecision __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setw __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, basic_string_view< _CharT, _Traits > __str)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _Tp, typename _CharT, class _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)
- template<typename _Traits>
  basic_ostream< char, _Traits > & **operator**<< (basic_ostream< char, _Traits > &, char16_t)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & **operator**<< (basic_ostream< char, _Traits > &, char32_t)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & **operator**<< (basic_ostream< char, _Traits > &, const char16_t *)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & **operator**<< (basic_ostream< char, _Traits > &, const char32_t *)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & **operator**<< (basic_ostream< char, _Traits > &, const wchar_t *)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & **operator**<< (basic_ostream< char, _Traits > &, wchar_t)=delete
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & **operator**<< (basic_ostream< wchar_t, _Traits > &, char16_t)=delete
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & **operator**<< (basic_ostream< wchar_t, _Traits > &, char32_t)=delete

- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & **operator**<< (basic_ostream< wchar_t, _Traits > &, const char16_t *)=delete
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & **operator**<< (basic_ostream< wchar_t, _Traits > &, const char32_t *)=delete
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **operator**<< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **operator**<< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__shift_left, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__shift_left, typename _Dom1::value_type >::result_type > **operator**<< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **operator**<< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **operator**<< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template ostream & **operator**<< (ostream &, _Resetiosflags)
- template ostream & **operator**<< (ostream &, _Setbase)
- template ostream & **operator**<< (ostream &, _Setfill< char >)
- template ostream & **operator**<< (ostream &, _Setiosflags)
- template ostream & **operator**<< (ostream &, _Setprecision)
- template ostream & **operator**<< (ostream &, _Setw)
- template ostream & **operator**<< (ostream &, const complex< double > &)
- template ostream & **operator**<< (ostream &, const complex< float > &)
- template ostream & **operator**<< (ostream &, const complex< long double > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)
- template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const discard_block_engine< _RandomNumberEngine, __p, __r > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const discrete_distribution< _IntType > &__x)

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const fisher_f_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const gamma_distribution< _RealType > &__x)
- template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const lognormal_distribution< _RealType > &__x)
- template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _↩
  UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const negative_binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const normal_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_constant_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_linear_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const poisson_distribution< _IntType > &__x)
- template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine, __k > &__x)
- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)
- template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const student_t_distribution< _RealType > &__x)
- template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)
- template wostream & **operator**<< (wostream &, _Resetiosflags)
- template wostream & **operator**<< (wostream &, _Setbase)
- template wostream & **operator**<< (wostream &, _Setfill< wchar_t >)
- template wostream & **operator**<< (wostream &, _Setiosflags)
- template wostream & **operator**<< (wostream &, _Setprecision)
- template wostream & **operator**<< (wostream &, _Setw)
- template wostream & **operator**<< (wostream &, const complex< double > &)
- template wostream & **operator**<< (wostream &, const complex< float > &)
- template wostream & **operator**<< (wostream &, const complex< long double > &)
- template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>
  std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)
- template<typename _CharT, typename _Traits, typename _Tp, typename _Dp>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const unique_ptr< _Tp, _Dp > &__p)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **operator**<= (const _Expr< _↩ Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **operator**<= (const _Expr< _↩ Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↩ __less_equal, typename _Dom1::value_type >::result_type > **operator**<= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
  constexpr bool **operator**<= (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() >= __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator**<= (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)

- template<typename _Tp, typename _Seq>
  bool operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **operator<=** (const typename ↩ _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **operator<=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp, typename _Dp>
  constexpr bool operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- strong_ordering operator<=> (const error_code &__lhs, const error_code &__rhs) noexcept
- strong_ordering operator<=> (const error_condition &__lhs, const error_condition &__rhs) noexcept
- template<typename _CharT, typename _Traits>
  constexpr auto **operator<=>** (basic_string_view< _CharT, _Traits > __x, type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
- template<typename _Tp, typename _Up, _Lock_policy _Lp>
  strong_ordering **operator<=>** (const __shared_ptr< _Tp, _Lp > &__a, const __shared_ptr< _Up, _Lp > &__b) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  strong_ordering **operator<=>** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp, size_t _Nm>
  constexpr __detail::__synth3way_t< _Tp > **operator<=>** (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr auto operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr auto operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
- template<typename _Tp, typename _Alloc>
  __detail::__synth3way_t< _Tp > operator<=> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  __detail::__synth3way_t< _Tp > operator<=> (const forward_list< _Tp, _Alloc > &__x, const forward_list< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  __detail::__synth3way_t< _Tp > operator<=> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< pair< const _Key, _Tp > > operator<=> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<three_way_comparable _Iterator>
  constexpr compare_three_way_result_t< _Iterator > **operator<=>** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
  constexpr compare_three_way_result_t< _IteratorL, _IteratorR > **operator<=>** (const move_iterator< _↩ IteratorL > &__x, const move_iterator< _IteratorR > &__y)

- template<typename _Tp, three_way_comparable _Seq>
  compare_three_way_result_t< _Seq > **operator<=>** (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<three_way_comparable _Iterator>
  constexpr compare_three_way_result_t< _Iterator, _Iterator > **operator<=>** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
  constexpr compare_three_way_result_t< _IteratorL, _IteratorR > **operator<=>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Tp, three_way_comparable _Seq>
  compare_three_way_result_t< _Seq > **operator<=>** (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename... _Tps, typename... _Ups>
  constexpr common_comparison_category_t< __detail::__synth3way_t< _Tps, _Ups >... > **operator<=>** (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer, typename unique_ptr< _Up, _Ep >::pointer > **operator<=>** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer > **operator<=>** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > operator<=> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- constexpr strong_ordering **operator<=>** (monostate, monostate) noexcept
- template<typename _T1, typename _T2>
  constexpr bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept
- bool operator== (const error_code &__lhs, const error_code &__rhs) noexcept
- bool operator== (const error_code &__lhs, const error_condition &__rhs) noexcept
- bool operator== (const error_condition &__lhs, const error_condition &__rhs) noexcept
- template<typename _Bi_iter, typename _Alloc>
  bool operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &__m2)
- template<typename _CharT, typename _Traits>
  constexpr bool **operator==** (basic_string_view< _CharT, _Traits > __x, type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  bool **operator==** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>
  bool **operator==** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **operator==** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **operator==** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__equal_to, typename _Dom1::value_type >::result_type > **operator==** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)

- template<typename _Tp, std::size_t _Nm>
  constexpr bool **operator==** (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr bool operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept
- template<typename _Tp, typename _Alloc>
  bool operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _StateT>
  bool operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)
- template<typename _Res, typename... _Args>
  bool operator== (const function< _Res(_Args...)> &__f, nullptr_t) noexcept
- template<typename _CharT, typename _Traits>
  bool **operator==** (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)
- template<typename _Tp, typename _Alloc>
  bool operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Iterator>
  constexpr bool **operator==** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator==** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Iterator>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator==** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _RealType>
  bool **operator==** (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)
- template<typename... _Tps, typename... _Ups>
  constexpr bool **operator==** (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)

- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **operator==** (const typename _↩ Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  bool **operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  bool **operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_↩ multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  bool **operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  bool **operator==** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **operator==** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp, typename _Alloc>
  constexpr bool operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- constexpr bool **operator==** (monostate, monostate) noexcept
- template<typename _OutA1, typename _OutA2, typename... _InA>
  bool **operator==** (const scoped_allocator_adaptor< _OutA1, _InA... > &__a, const scoped_allocator_adaptor< _OutA2, _InA... > &__b) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename ↩ __fun< struct std::__greater, typename _Dom::value_type >::result_type > **operator>** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > **operator>** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__↩ greater, typename _Dom1::value_type >::result_type > **operator>** (const _Expr< _Dom1, typename _Dom1::↩ value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
  constexpr bool **operator>** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)

- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &↩ __y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > **operator>** (const typename _↩ Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > **operator>** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp, typename _Dp>
  constexpr bool operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **operator>=** (const ↩ _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **operator>=** (const ↩ _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↩ __greater_equal, typename _Dom1::value_type >::result_type > **operator>=** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator>=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() <= __y.base() } -> convertible_to<bool>; }
  constexpr bool **operator>=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Tp, typename _Seq>
  bool operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)

- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **operator>=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **operator>=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp, typename _Dp>
  bool operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Istream, typename _Tp>
  __rvalue_stream_extraction_t< _Istream, _Tp > operator>> (_Istream &&__is, _Tp &&__x)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT, typename _Traits, typename _MoneyT>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Get_money< _↩ MoneyT > __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Get_time< _CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Resetiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setbase __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setfill< _CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setprecision __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **operator>>** (basic_istream< _CharT, _Traits > &__is, _Setw __f)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _↩ CharT, _Traits, _Alloc > &__str)
- template<typename _Tp, typename _CharT, class _Traits>
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)
- template<> basic_istream< char > & **operator>>** (basic_istream< char > &__is, basic_string< char > &_↩ _str)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **operator>>** (const _Expr< _↩ Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **operator>>** (const _Expr< _↩ Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)

- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↩
  __shift_right, typename _Dom1::value_type >::result_type > **operator**>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **operator**>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **operator**>> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template istream & **operator**>> (istream &, _Resetiosflags)
- template istream & **operator**>> (istream &, _Setbase)
- template istream & **operator**>> (istream &, _Setfill< char >)
- template istream & **operator**>> (istream &, _Setiosflags)
- template istream & **operator**>> (istream &, _Setprecision)
- template istream & **operator**>> (istream &, _Setw)
- template istream & **operator**>> (istream &, complex< double > &)
- template istream & **operator**>> (istream &, complex< float > &)
- template istream & **operator**>> (istream &, complex< long double > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, chi_squared_distribution< _RealType > &__x)
- template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, discard_block_engine< _RandomNumberEngine, __p, __r > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, discrete_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, fisher_f_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, gamma_distribution< _RealType > &__x)
- template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, lognormal_distribution< _RealType > &__x)

- template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, negative_binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, normal_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, piecewise_constant_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, piecewise_linear_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, poisson_distribution< _IntType > &__x)
- template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, shuffle_order_engine< _RandomNumberEngine, __k > &__x)
- template<typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::bernoulli_distribution &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, student_t_distribution< _RealType > &__x)
- template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &__is, subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)
- template wistream & **operator**>> (wistream &, _Resetiosflags)
- template wistream & **operator**>> (wistream &, _Setbase)
- template wistream & **operator**>> (wistream &, _Setfill< wchar_t >)
- template wistream & **operator**>> (wistream &, _Setiosflags)
- template wistream & **operator**>> (wistream &, _Setprecision)
- template wistream & **operator**>> (wistream &, _Setw)

- template wistream & **operator**>> (wistream &, complex< double > &)
- template wistream & **operator**>> (wistream &, complex< float > &)
- template wistream & **operator**>> (wistream &, complex< long double > &)
- constexpr _Ios_Fmtflags **operator**^ (_Ios_Fmtflags __a, _Ios_Fmtflags __b) noexcept
- constexpr _Ios_Iostate **operator**^ (_Ios_Iostate __a, _Ios_Iostate __b) noexcept
- constexpr _Ios_Openmode **operator**^ (_Ios_Openmode __a, _Ios_Openmode __b) noexcept
- constexpr chars_format **operator**^ (chars_format __lhs, chars_format __rhs) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **operator**^ (const _Expr< _↩ Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **operator**^ (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↩ __bitwise_xor, typename _Dom1::value_type >::result_type > **operator**^ (const _Expr< _Dom1, typename _↩ Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **operator**^ (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **operator**^ (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr launch **operator**^ (launch __x, launch __y) noexcept
- constexpr const _Ios_Fmtflags & **operator**^= (_Ios_Fmtflags &__a, _Ios_Fmtflags __b) noexcept
- constexpr const _Ios_Iostate & **operator**^= (_Ios_Iostate &__a, _Ios_Iostate __b) noexcept
- constexpr const _Ios_Openmode & **operator**^= (_Ios_Openmode &__a, _Ios_Openmode __b) noexcept
- constexpr chars_format & **operator**^= (chars_format &__lhs, chars_format __rhs) noexcept
- constexpr launch & **operator**^= (launch &__x, launch __y) noexcept
- constexpr _Ios_Fmtflags **operator**| (_Ios_Fmtflags __a, _Ios_Fmtflags __b) noexcept
- constexpr _Ios_Iostate **operator**| (_Ios_Iostate __a, _Ios_Iostate __b) noexcept
- constexpr _Ios_Openmode **operator**| (_Ios_Openmode __a, _Ios_Openmode __b) noexcept
- constexpr chars_format **operator**| (chars_format __lhs, chars_format __rhs) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **operator**| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **operator**| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↩ __bitwise_or, typename _Dom1::value_type >::result_type > **operator**| (const _Expr< _Dom1, typename _↩ Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **operator**| (const typename _↩ Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **operator**| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr launch **operator**| (launch __x, launch __y) noexcept
- constexpr memory_order **operator**| (memory_order __m, __memory_order_modifier __mod) noexcept
- constexpr const _Ios_Fmtflags & **operator**|**=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b) noexcept
- constexpr const _Ios_Iostate & **operator**|**=** (_Ios_Iostate &__a, _Ios_Iostate __b) noexcept
- constexpr const _Ios_Openmode & **operator**|**=** (_Ios_Openmode &__a, _Ios_Openmode __b) noexcept
- constexpr chars_format & **operator**|**=** (chars_format &__lhs, chars_format __rhs) noexcept
- constexpr launch & **operator**|**=** (launch &__x, launch __y) noexcept
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **operator**|| (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **operator**|| (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↩ __logical_or, typename _Dom1::value_type >::result_type > **operator**|| (const _Expr< _Dom1, typename _↩ Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **operator**|| (const typename _↩ Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **operator**|| (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr _Ios_Fmtflags **operator**~ (_Ios_Fmtflags __a) noexcept
- constexpr _Ios_Iostate **operator**~ (_Ios_Iostate __a) noexcept
- constexpr _Ios_Openmode **operator**~ (_Ios_Openmode __a) noexcept
- constexpr chars_format **operator**~ (chars_format __fmt) noexcept
- constexpr launch **operator**~ (launch __x) noexcept
- template<typename _Pointer = void, typename _Smart, typename... _Args>
  auto out_ptr (_Smart &__s, _Args &&... __args)
- template<typename _Fun, typename _Signature = __function_guide_t<_Fun, decltype(&_Fun::operator())>>
  packaged_task (_Fun) -> packaged_task< _Signature >
- template<typename _Res, typename... _ArgTypes>
  packaged_task (_Res(∗)(_ArgTypes...)) -> packaged_task< _Res(_ArgTypes...)>
- template<typename _RAIter>
  constexpr void **partial_sort** (_RAIter, _RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **partial_sort** (_RAIter, _RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _Random↩ AccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _Random↩ AccessIterator __last, _Compare __comp)

- template<typename _IIter, typename _RAIter>
  constexpr _RAIter **partial_sort_copy** (_IIter, _IIter, _RAIter, _RAIter)
- template<typename _IIter, typename _RAIter, typename _Compare>
  constexpr _RAIter **partial_sort_copy** (_IIter, _IIter, _RAIter, _RAIter, _Compare)
- template<typename _InputIterator, typename _RandomAccessIterator>
  constexpr _RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _Random↩
  AccessIterator __result_first, _RandomAccessIterator __result_last)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare>
  constexpr _RandomAccessIterator partial_sort_copy (_InputIterator __first, _InputIterator __last, _Random↩
  AccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator partial_sum (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _↩
  BinaryOperation __binary_op)
- template<typename _BIter, typename _Predicate>
  constexpr _BIter **partition** (_BIter, _BIter, _Predicate)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _IIter, typename _OIter1, typename _OIter2, typename _Predicate>
  constexpr pair< _OIter1, _OIter2 > **partition_copy** (_IIter, _IIter, _OIter1, _OIter2, _Predicate)
- template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate>
  constexpr pair< _OutputIterator1, _OutputIterator2 > partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)
- template<typename _FIter, typename _Predicate>
  constexpr _FIter **partition_point** (_FIter, _FIter, _Predicate)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _Tp>
  complex< _Tp > polar (const _Tp &, const _Tp &=_Tp(0))
- template<typename _RAIter>
  constexpr void **pop_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **pop_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void pop_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<__simd_unsigned_integer _Vp>
  constexpr rebind_t< make_signed_t< typename _Vp::value_type >, _Vp > **popcount** (const _Vp &__v) noexcept
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **pow** (_Tp __x, _Up __y)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _↩
  Dom::value_type > **pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::↩
  value_type &__t)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _↩
  Dom::value_type > **pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)

- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::_Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > **pow** (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::↩ value_type > &__e2)
- template<typename _Tp>
  complex< _Tp > pow (const _Tp &, const complex< _Tp > &)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **pow** (const _Tp &__x, const std::complex< _Up > &__y)
- template<typename _Tp>
  complex< _Tp > pow (const complex< _Tp > &, const _Tp &)
- template<typename _Tp>
  complex< _Tp > pow (const complex< _Tp > &, const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > pow (const complex< _Tp > &, int)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **pow** (const std::complex< _Tp > &__x, const _Up &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _↩ Dom::value_type > **pow** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::↩ value_type > &__e)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > **pow** (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > **pow** (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > **pow** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩ Dom::value_type > **pow** (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- constexpr float **pow** (float __x, float __y)
- constexpr long double **pow** (long double __x, long double __y)
- template<typename _BidirectionalIterator>
  constexpr _BidirectionalIterator **prev** (_BidirectionalIterator __x, typename iterator_traits< _BidirectionalIterator >::difference_type __n=1)
- template<typename _BidirectionalIterator>
  constexpr bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _BIter>
  constexpr bool **prev_permutation** (_BIter, _BIter)
- template<typename _BIter, typename _Compare>
  constexpr bool **prev_permutation** (_BIter, _BIter, _Compare)

- template<typename _Compare, typename _Container, typename = _RequireNotAllocator<_Compare>, typename = _RequireNot↩ Allocator<_Container>>
  priority_queue (_Compare, _Container) -> priority_queue< typename _Container::value_type, _Container, _↩ Compare >
- template<typename _Compare, typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>
  priority_queue (_Compare, _Container, _Allocator) -> priority_queue< typename _Container::value_type, _↩ Container, _Compare >
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Compare = less<_ValT>, typename _Container = vector<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNot↩ Allocator<_Compare>, typename = _RequireNotAllocator<_Container>>
  priority_queue (_InputIterator, _InputIterator, _Compare=_Compare(), _Container=_Container()) -> priority_↩ queue< _ValT, _Container, _Compare >
- template<typename _Tp>
  std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **proj** (_Tp __x)
- template<__simd_complex _Vp>
  constexpr _Vp **proj** (const _Vp &__x) noexcept
- template<typename _Tp>
  std::complex< _Tp > **proj** (const std::complex< _Tp > &)
- template<typename _Arg, typename _Result>
  pointer_to_unary_function< _Arg, _Result > ptr_fun (_Result(*__x)(_Arg))
- template<typename _Arg1, typename _Arg2, typename _Result>
  pointer_to_binary_function< _Arg1, _Arg2, _Result > ptr_fun (_Result(*__x)(_Arg1, _Arg2))
- template<typename _RAIter>
  constexpr void **push_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **push_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void push_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _MoneyT>
  _Put_money< _MoneyT > put_money (const _MoneyT &__mon, bool __intl=false)
- template<typename _CharT>
  _Put_time< _CharT > put_time (const std::tm *__tmb, const _CharT *__fmt)
- template<typename _Container, typename = _RequireNotAllocator<_Container>>
  queue (_Container) -> queue< typename _Container::value_type, _Container >
- template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>
  queue (_Container, _Allocator) -> queue< typename _Container::value_type, _Container >
- template<typename _RAIter>
  void **random_shuffle** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Generator>
  void **random_shuffle** (_RAIter, _RAIter, _Generator &&)
- template<typename _RandomAccessIterator>
  void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _RandomNumberGenerator>
  void random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumber↩ Generator &&__rand)
- template<typename _Container>
  constexpr auto rbegin (_Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__cont.rbegin())
- template<typename _Tp, size_t _Nm>
  constexpr reverse_iterator< _Tp * > rbegin (_Tp(&__arr)[_Nm]) noexcept

- template<typename _Container>
  constexpr auto rbegin (const _Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__cont.↩ rbegin())
- template<typename _Tp>
  constexpr reverse_iterator< const _Tp * > rbegin (initializer_list< _Tp > __il) noexcept
- constexpr _RealSimd **real** () const noexcept
- template<typename _Tp>
  constexpr __gnu_cxx::__promote< _Tp >::__type **real** (_Tp __x)
- template<typename _Tp>
  constexpr _Tp **real** (const complex< _Tp > &__z)
- template<typename _InputIterator>
  constexpr iterator_traits< _InputIterator >::value_type reduce (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator, typename _Tp>
  constexpr _Tp reduce (_InputIterator __first, _InputIterator __last, _Tp __init)
- template<typename _InputIterator, typename _Tp, typename _BinaryOperation>
  constexpr _Tp reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __binary_op)
- template<typename _Tp>
  reference_wrapper (_Tp &) -> reference_wrapper< _Tp >
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > **reinterpret_pointer_cast** (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **remainder** (_Tp __x, _Up __y)
- constexpr float **remainder** (float __x, float __y)
- constexpr long double **remainder** (long double __x, long double __y)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr _FIter **remove** (_FIter, _FIter, const _Tp &)
- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _IIter, typename _OIter, typename _Tp = typename iterator_traits<_IIter>::value_type>
  constexpr _OIter **remove_copy** (_IIter, _IIter, _OIter, const _Tp &)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)
- template<typename _IIter, typename _OIter, typename _Predicate>
  constexpr _OIter **remove_copy_if** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
  constexpr _OutputIterator remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred)
- template<typename _FIter, typename _Predicate>
  constexpr _FIter **remove_if** (_FIter, _FIter, _Predicate)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _Tp, typename _Up>
  __gnu_cxx::__promote_2< _Tp, _Up >::__type **remquo** (_Tp __x, _Up __y, int *__pquo)
- float **remquo** (float __x, float __y, int *__pquo)
- long double **remquo** (long double __x, long double __y, int *__pquo)
- template<typename _Container>
  constexpr auto rend (_Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.rend())
- template<typename _Tp, size_t _Nm>
  constexpr reverse_iterator< _Tp * > rend (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto rend (const _Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.rend())

- template<typename _Tp>
  constexpr reverse_iterator< const _Tp * > rend (initializer_list< _Tp > __il) noexcept
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr void **replace** (_FIter, _FIter, const _Tp &, const _Tp &)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _IIter, typename _OIter, typename _Tp>
  constexpr _OIter **replace_copy** (_IIter, _IIter, _OIter, const _Tp &, const _Tp &)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator replace_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>
  constexpr _OutputIterator replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Predicate __pred, const _Tp &__new_value)
- template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp = typename iterator_traits<_OIter>::value_type>
  constexpr _OIter **replace_copy_if** (_Iter, _Iter, _OIter, _Predicate, const _Tp &)
- template<typename _FIter, typename _Predicate, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr void **replace_if** (_FIter, _FIter, _Predicate, const _Tp &)
- template<typename _ForwardIterator, typename _Predicate, typename _Tp>
  constexpr void replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)
- _Resetiosflags resetiosflags (ios_base::fmtflags __mask)
- void rethrow_exception (exception_ptr)
- template<typename _Ex>
  void rethrow_if_nested (const _Ex &__ex)
- template<typename _Tp>
  void return_temporary_buffer (_Tp *__p)
- template<typename _BidirectionalIterator>
  constexpr void reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BIter>
  constexpr void **reverse** (_BIter, _BIter)
- template<typename _BidirectionalIterator, typename _OutputIterator>
  constexpr _OutputIterator reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _OutputIterator __result)
- template<typename _BIter, typename _OIter>
  constexpr _OIter **reverse_copy** (_BIter, _BIter, _OIter)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type riemann_zeta (_Tp __s)
- float riemann_zetaf (float __s)
- long double riemann_zetal (long double __s)
- ios_base & right (ios_base &__base)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **rint** (_Tp __x)
- constexpr float **rint** (float __x)
- constexpr long double **rint** (long double __x)
- template<typename _FIter>
  constexpr _FIter **rotate** (_FIter, _FIter, _FIter)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last)
- template<typename _FIter, typename _OIter>
  constexpr _OIter **rotate_copy** (_FIter, _FIter, _FIter, _OIter)

- template<typename _ForwardIterator, typename _OutputIterator>
  constexpr _OutputIterator rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator _↩
  _last, _OutputIterator __result)
- template<__simd_unsigned_integer _V0, __simd_integral _V1>
  constexpr _V0 **rotl** (const _V0 &__v, const _V1 &__s) noexcept
- template<__simd_unsigned_integer _V0, __simd_integral _V1>
  constexpr _V0 **rotr** (const _V0 &__v, const _V1 &__s) noexcept
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **round** (_Tp __x)
- constexpr float **round** (float __x)
- constexpr long double **round** (long double __x)
- template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomBitGenerator>
  _SampleIterator sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n, _UniformRandomBitGenerator &&__g)
- template<typename _Tp>
  constexpr _Tp saturating_add (_Tp __x, _Tp __y) noexcept
- template<typename _Res, typename _Tp>
  constexpr _Res saturating_cast (_Tp __x) noexcept
- template<typename _Tp>
  constexpr _Tp saturating_div (_Tp __x, _Tp __y) noexcept
- template<typename _Tp>
  constexpr _Tp saturating_mul (_Tp __x, _Tp __y) noexcept
- template<typename _Tp>
  constexpr _Tp saturating_sub (_Tp __x, _Tp __y) noexcept
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **scalbln** (_Tp __x, long __ex)
- constexpr float **scalbln** (float __x, long __ex)
- constexpr long double **scalbln** (long double __x, long __ex)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **scalbn** (_Tp __x, int __ex)
- constexpr float **scalbn** (float __x, int __ex)
- constexpr long double **scalbn** (long double __x, int __ex)
- ios_base & scientific (ios_base &__base)
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _ForwardIterator, typename _Searcher>
  constexpr _ForwardIterator search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__↩
  searcher)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 ↩
  __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 ↩
  __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)
- template<typename _FIter, typename _Size, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr _FIter **search_n** (_FIter, _FIter, _Size, const _Tp &)
- template<typename _FIter, typename _Size, typename _Tp = typename iterator_traits<_FIter>::value_type, typename _BinaryPredicate>
  constexpr _FIter **search_n** (_FIter, _FIter, _Size, const _Tp &, _BinaryPredicate)

- template<typename _ForwardIterator, typename _Integer, typename _Tp>
  constexpr _ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)
- template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
  constexpr _ForwardIterator search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  set (initializer_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  set (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _Compare, _Allocator >
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- new_handler set_new_handler (new_handler) throw ()
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- terminate_handler set_terminate (terminate_handler) noexcept

- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↩
  _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, ↩
  _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- _Setbase setbase (int __base)
- template<typename _CharT>
  _Setfill< _CharT > setfill (_CharT __c)
- _Setiosflags setiosflags (ios_base::fmtflags __mask)
- _Setprecision setprecision (int __n)
- _Setw setw (int __n)
- template<typename _Tp, typename _Del>
  shared_ptr (unique_ptr< _Tp, _Del >) -> shared_ptr< _Tp >
- template<typename _Tp>
  shared_ptr (weak_ptr< _Tp >) -> shared_ptr< _Tp >
- template<typename _ForwardIterator>
  constexpr _ForwardIterator **shift_left** (_ForwardIterator __first, _ForwardIterator __last, typename iterator_traits< _ForwardIterator >::difference_type __n)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator **shift_right** (_ForwardIterator __first, _ForwardIterator __last, typename iterator_traits< _ForwardIterator >::difference_type __n)
- ios_base & showbase (ios_base &__base)
- ios_base & showpoint (ios_base &__base)
- ios_base & showpos (ios_base &__base)
- template<typename _RAIter, typename _UGenerator>
  void **shuffle** (_RAIter, _RAIter, _UGenerator &&)
- template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator>
  void shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumberGenerator &&__g)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **signbit** (_Tp __x)
- constexpr bool **signbit** (double __x)
- constexpr bool **signbit** (float __x)
- constexpr bool **signbit** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **sin** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Sin, _Expr, _Dom >, typename _Dom::value_type > **sin** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > sin (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Sin, _ValArray, _Tp >, _Tp > **sin** (const valarray< _Tp > &__v)
- constexpr float **sin** (float __x)
- constexpr long double **sin** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **sinh** (_Tp __x)

- template<class _Dom>
  _Expr< _UnClos< struct std::_Sinh, _Expr, _Dom >, typename _Dom::value_type > **sinh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > sinh (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Sinh, _ValArray, _Tp >, _Tp > **sinh** (const valarray< _Tp > &__v)
- constexpr float **sinh** (float __x)
- constexpr long double **sinh** (long double __x)
- template<typename _Container>
  constexpr auto size (const _Container &__cont) noexcept(noexcept(__cont.size())) -> decltype(__cont.size())
- template<typename _Tp, size_t _Nm>
  constexpr size_t size (const _Tp(&)[_Nm]) noexcept
- ios_base & skipws (ios_base &__base)
- template<typename _RAIter>
  constexpr void **sort** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **sort** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RAIter>
  constexpr void **sort_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **sort_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void sort_heap (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type sph_bessel (unsigned int __n, _Tp __x)
- float sph_besself (unsigned int __n, float __x)
- long double sph_bessell (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float sph_legendref (unsigned int __l, unsigned int __m, float __theta)
- long double sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type sph_neumann (unsigned int __n, _Tp __x)
- float sph_neumannf (unsigned int __n, float __x)
- long double sph_neumannl (unsigned int __n, long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **sqrt** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Sqrt, _Expr, _Dom >, typename _Dom::value_type > **sqrt** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > sqrt (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Sqrt, _ValArray, _Tp >, _Tp > **sqrt** (const valarray< _Tp > &__v)
- constexpr float **sqrt** (float __x)

- constexpr long double **sqrt** (long double __x)
- template<typename _Container>
  constexpr auto ssize (const _Container &__cont) noexcept(noexcept(__cont.size())) -> common_type_t< ptrdiff_t, make_signed_t< decltype(__cont.size())> >
- template<typename _Tp, ptrdiff_t _Num>
  constexpr ptrdiff_t ssize (const _Tp(&)[_Num]) noexcept
- template<typename _BIter, typename _Predicate>
  constexpr _BIter **stable_partition** (_BIter, _BIter, _Predicate)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _RAIter>
  constexpr void **stable_sort** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **stable_sort** (_RAIter, _RAIter, _Compare)
- template<typename _RandomAccessIterator>
  constexpr void stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _Container, typename = _RequireNotAllocator<_Container>>
  stack (_Container) -> stack< typename _Container::value_type, _Container >
- template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>
  stack (_Container, _Allocator) -> stack< typename _Container::value_type, _Container >
- template<typename _Tp>
  const _Tp * **start_lifetime_as** (const void *__p) noexcept
- template<typename _Tp>
  const volatile _Tp * **start_lifetime_as** (const volatile void *__p) noexcept
- template<typename _Tp>
  _Tp * **start_lifetime_as** (void *__p) noexcept
- template<typename _Tp>
  volatile _Tp * **start_lifetime_as** (volatile void *__p) noexcept
- template<typename _Tp>
  const _Tp * **start_lifetime_as_array** (const void *__p, size_t __n) noexcept
- template<typename _Tp>
  const volatile _Tp * **start_lifetime_as_array** (const volatile void *__p, size_t __n) noexcept
- template<typename _Tp>
  _Tp * **start_lifetime_as_array** (void *__p, size_t __n) noexcept
- template<typename _Tp>
  volatile _Tp * **start_lifetime_as_array** (volatile void *__p, size_t __n) noexcept
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- double **stod** (const string &__str, size_t *__idx=0)
- double **stod** (const wstring &__str, size_t *__idx=0)
- float **stof** (const string &__str, size_t *__idx=0)
- float **stof** (const wstring &__str, size_t *__idx=0)
- int **stoi** (const string &__str, size_t *__idx=0, int __base=10)
- int **stoi** (const wstring &__str, size_t *__idx=0, int __base=10)
- long **stol** (const string &__str, size_t *__idx=0, int __base=10)
- long **stol** (const wstring &__str, size_t *__idx=0, int __base=10)
- long double **stold** (const string &__str, size_t *__idx=0)
- long double **stold** (const wstring &__str, size_t *__idx=0)
- long long **stoll** (const string &__str, size_t *__idx=0, int __base=10)
- long long **stoll** (const wstring &__str, size_t *__idx=0, int __base=10)
- unsigned long **stoul** (const string &__str, size_t *__idx=0, int __base=10)
- unsigned long **stoul** (const wstring &__str, size_t *__idx=0, int __base=10)

- unsigned long long **stoull** (const string &__str, size_t *__idx=0, int __base=10)
- unsigned long long **stoull** (const wstring &__str, size_t *__idx=0, int __base=10)
- char * **strchr** (char *__s, int __n)
- char * **strpbrk** (char *__s1, const char *__s2)
- char * **strrchr** (char *__s, int __n)
- char * **strstr** (char *__s1, const char *__s2)
- template<typename _IndexType, size_t... _Extents, typename... _RawSlices>
  constexpr auto **subextents** (const extents< _IndexType, _Extents... > &__exts, _RawSlices... __raw_slices)
- template<typename _ElementType, typename _Extents, typename _Layout, typename _Accessor, typename... _RawSlices>
  constexpr auto **submdspan** (const mdspan< _ElementType, _Extents, _Layout, _Accessor > &__md, _Raw↩
  Slices... __raw_slices)
- template<typename _Ch_type, typename _Rx_traits>
  void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs) noex-
  cept
- template<typename _Bi_iter, typename _Alloc>
  void swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  void **swap** (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  void **swap** (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept
- template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc>
  void **swap** (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOf↩
  Value, _Compare, _Alloc > &__y)
- template<typename _Tp>
  constexpr _Require< __not_< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable<
  _Tp > > swap (_Tp &, _Tp &) noexcept(__and_< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable<
  _Tp > >::value)
- template<typename _Tp>
  constexpr void swap (_Tp &__a, _Tp &__b) noexcept(/*conditional */) is_nothrow_move_assignable< _Tp > >
- template<typename _Tp, size_t _Nm>
  constexpr void swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(/*conditional */)
- template<typename _Tp, size_t _Nm>
  constexpr __enable_if_t< __is_swappable< _Tp >::value > swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]↩
) noexcept(__is_nothrow_swappable< _Tp >::value)
- template<typename _Tp, std::size_t _Nm>
  __enable_if_t<!__array_traits< _Tp, _Nm >::_Is_swappable::value > **swap** (array< _Tp, _Nm > &, array< _Tp,
  _Nm > &)=delete
- template<typename _Tp, std::size_t _Nm>
  constexpr __enable_if_t< __array_traits< _Tp, _Nm >::_Is_swappable::value > **swap** (array< _Tp, _Nm > &↩
  __one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.swap(__two)))
- template<class _CharT, class _Traits>
  void swap (basic_filebuf< _CharT, _Traits > &__x, basic_filebuf< _CharT, _Traits > &__y)
- template<class _CharT, class _Traits>
  void swap (basic_fstream< _CharT, _Traits > &__x, basic_fstream< _CharT, _Traits > &__y)
- template<class _CharT, class _Traits>
  void swap (basic_ifstream< _CharT, _Traits > &__x, basic_ifstream< _CharT, _Traits > &__y)
- template<class _CharT, class _Traits, class _Allocator>
  void swap (basic_istringstream< _CharT, _Traits, _Allocator > &__x, basic_istringstream< _CharT, _Traits, _↩
  Allocator > &__y)
- template<class _CharT, class _Traits>
  void swap (basic_ofstream< _CharT, _Traits > &__x, basic_ofstream< _CharT, _Traits > &__y)

- template<class _CharT, class _Traits, class _Allocator>
  void swap (basic_ostringstream< _CharT, _Traits, _Allocator > &__x, basic_ostringstream< _CharT, _Traits, _Allocator > &__y)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr void swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept(/*conditional */)
- template<class _CharT, class _Traits, class _Allocator>
  void swap (basic_stringbuf< _CharT, _Traits, _Allocator > &__x, basic_stringbuf< _CharT, _Traits, _Allocator > &__y) noexcept(noexcept(__x.swap(__y)))
- template<class _CharT, class _Traits, class _Allocator>
  void swap (basic_stringstream< _CharT, _Traits, _Allocator > &__x, basic_stringstream< _CharT, _Traits, _Allocator > &__y)
- template<typename _Tp, typename _Alloc>
  void swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y) noexcept(/*conditional */)
- constexpr void **swap** (exception_ptr &__lhs, exception_ptr &__rhs)
- template<typename _Tp, typename _Alloc>
  void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))
- template<typename _Res, typename... _Args>
  void swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y) noexcept
- template<typename _Tp, typename _Alloc>
  void swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y) noexcept(/*conditional */)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  void swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  void swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)
- template<typename _Key, typename _Compare, typename _Alloc>
  void swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)
- template<typename _Res, typename... _ArgTypes>
  void swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y) noexcept
- template<typename _Tp, typename _Sequence, typename _Compare>
  constexpr enable_if< __and_< __is_swappable< _Sequence >, __is_swappable< _Compare > >::value >::type **swap** (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Res>
  void **swap** (promise< _Res > &__x, promise< _Res > &__y) noexcept
- template<typename _Tp, typename _Seq>
  enable_if< __is_swappable< _Seq >::value >::type **swap** (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Key, typename _Compare, typename _Alloc>
  void swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)
- template<typename _Tp, typename _Seq>
  enable_if< __is_swappable< _Seq >::value >::type **swap** (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename... _Elements>
  constexpr enable_if<!__and_< __is_swappable< _Elements >... >::value >::type swap (tuple< _Elements... > &, tuple< _Elements... > &)=delete
- template<typename... _Elements>
  constexpr enable_if< __and_< __is_swappable< _Elements >... >::value >::type swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))

- template<typename _Tp, typename _Dp>
  enable_if<!__is_swappable< _Dp >::value >::type **swap** (unique_ptr< _Tp, _Dp > &, unique_ptr< _Tp, _Dp > &)=delete
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  void **swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  void **swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  void **swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  void **swap** (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp, typename _Alloc>
  constexpr void swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y) noexcept(/*conditional */)
- void **swap** (thread &__x, thread &__y) noexcept
- template<typename _Mutex>
  void swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y) noexcept
- template<typename _Tp, typename _Dp>
  constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept
- template<typename _Tp>
  void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter2 **swap_ranges** (_FIter1, _FIter1, _FIter2)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator2 swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)
- const error_category & system_category () noexcept
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **tan** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Tan, _Expr, _Dom >, typename _Dom::value_type > **tan** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > tan (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Tan, _ValArray, _Tp >, _Tp > **tan** (const valarray< _Tp > &__v)
- constexpr float **tan** (float __x)
- constexpr long double **tan** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **tanh** (_Tp __x)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Tanh, _Expr, _Dom >, typename _Dom::value_type > **tanh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  complex< _Tp > tanh (const complex< _Tp > &)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Tanh, _ValArray, _Tp >, _Tp > **tanh** (const valarray< _Tp > &__v)
- constexpr float **tanh** (float __x)

- constexpr long double **tanh** (long double __x)
- void terminate () noexcept
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **tgamma** (_Tp __x)
- constexpr float **tgamma** (float __x)
- constexpr long double **tgamma** (long double __x)
- template<typename _Tp>
  void throw_with_nested (_Tp &&__t)
- template<typename... _Elements>
  constexpr tuple< _Elements &... > tie (_Elements &... __args) noexcept
- template<typename _Tp>
  constexpr _Tp * to_address (_Tp *__ptr) noexcept
- template<typename _Ptr>
  constexpr auto to_address (const _Ptr &__ptr) noexcept
- to_chars_result **to_chars** (char *, char *, bool, int=10)=delete
- constexpr to_chars_result **to_chars** (char *__first, char *__last, char __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, signed char __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, signed int __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, signed long __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, signed long long __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, signed short __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, unsigned char __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, unsigned int __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, unsigned long __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, unsigned long long __value, int __base=10)
- constexpr to_chars_result **to_chars** (char *__first, char *__last, unsigned short __value, int __base=10)
- string **to_string** (double __val)
- string **to_string** (float __val)
- string **to_string** (int __val) noexcept
- string **to_string** (long __val) noexcept
- string **to_string** (long double __val)
- string **to_string** (long long __val)
- string **to_string** (unsigned __val) noexcept
- string **to_string** (unsigned long __val) noexcept
- string **to_string** (unsigned long long __val)
- wstring **to_wstring** (double __val)
- wstring **to_wstring** (float __val)
- wstring **to_wstring** (int __val)
- wstring **to_wstring** (long __val)
- wstring **to_wstring** (long double __val)
- wstring **to_wstring** (long long __val)
- wstring **to_wstring** (unsigned __val)
- wstring **to_wstring** (unsigned long __val)
- wstring **to_wstring** (unsigned long long __val)
- template<typename _CharT>
  _CharT tolower (_CharT __c, const locale &__loc)
- template<typename _CharT>
  _CharT toupper (_CharT __c, const locale &__loc)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  constexpr _OIter **transform** (_IIter, _IIter, _OIter, _UnaryOperation)

- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BinaryOperation>
  constexpr _OIter **transform** (_IIter1, _IIter1, _IIter2, _OIter, _BinaryOperation)
- template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation>
  constexpr _OutputIterator transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _↩
  UnaryOperation __unary_op)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _↩
  OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>
  constexpr _OutputIterator transform_exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init, _BinaryOperation __binary_op, _UnaryOperation __unary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation>
  constexpr _OutputIterator transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryOperation, typename _UnaryOperation, typename _Tp>
  constexpr _OutputIterator transform_inclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryOperation __binary_op, _UnaryOperation __unary_op, _Tp __init)
- template<typename _InputIterator, typename _Tp, typename _BinaryOperation, typename _UnaryOperation>
  constexpr _Tp transform_reduce (_InputIterator __first, _InputIterator __last, _Tp __init, _BinaryOperation __↩
  binary_op, _UnaryOperation __unary_op)
- template<typename _InputIterator1, typename _InputIterator2, typename _Tp>
  constexpr _Tp transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init)
- template<typename _InputIterator1, typename _InputIterator2, typename _Tp, typename _BinaryOperation1, typename _Binary↩
  Operation2>
  constexpr _Tp transform_reduce (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Tp __init, _BinaryOperation1 __binary_op1, _BinaryOperation2 __binary_op2)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **trunc** (_Tp __x)
- constexpr float **trunc** (float __x)
- constexpr long double **trunc** (long double __x)
- template<typename _L1, typename _L2, typename... _L3>
  int try_lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)
- template<typename... _UTypes>
  tuple (_UTypes...) -> tuple< _UTypes... >
- template<typename _Alloc, typename... _UTypes>
  tuple (allocator_arg_t, _Alloc, _UTypes...) -> tuple< _UTypes... >
- template<typename _Alloc, typename _T1, typename _T2>
  tuple (allocator_arg_t, _Alloc, pair< _T1, _T2 >) -> tuple< _T1, _T2 >
- template<typename _Alloc, typename... _UTypes>
  tuple (allocator_arg_t, _Alloc, tuple< _UTypes... >) -> tuple< _UTypes... >
- template<typename _T1, typename _T2>
  tuple (pair< _T1, _T2 >) -> tuple< _T1, _T2 >
- template<typename... _Tpls, typename = typename enable_if<__and_<__is_tuple_like<_Tpls>...>::value>::type>
  constexpr auto tuple_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls... >::__type
- bool uncaught_exception () noexcept
- template<typename _Tp, typename _Alloc, typename... _Args>
  constexpr _Tp * **uninitialized_construct_using_allocator** (_Tp *__p, const _Alloc &__a, _Args &&... __args)
- template<typename _InputIterator, typename _ForwardIterator>
  constexpr _ForwardIterator uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __↩
  result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator>
  constexpr _ForwardIterator uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)

- template<typename _ForwardIterator>
  constexpr void uninitialized_default_construct (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator uninitialized_default_construct_n (_ForwardIterator __first, _Size __count)
- template<typename _ForwardIterator, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>
  constexpr void uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)
- template<typename _ForwardIterator, typename _Size, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>
  constexpr _ForwardIterator uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)
- template<typename _InputIterator, typename _ForwardIterator>
  constexpr _ForwardIterator uninitialized_move (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator>
  constexpr pair< _InputIterator, _ForwardIterator > uninitialized_move_n (_InputIterator __first, _Size __count, _ForwardIterator __result)
- template<typename _ForwardIterator>
  constexpr void uninitialized_value_construct (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator uninitialized_value_construct_n (_ForwardIterator __first, _Size __count)
- template<typename _FIter>
  constexpr _FIter **unique** (_FIter, _FIter)
- template<typename _FIter, typename _BinaryPredicate>
  constexpr _FIter **unique** (_FIter, _FIter, _BinaryPredicate)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __binary_pred)
- template<typename _IIter, typename _OIter>
  constexpr _OIter **unique_copy** (_IIter, _IIter, _OIter)
- template<typename _IIter, typename _OIter, typename _BinaryPredicate>
  constexpr _OIter **unique_copy** (_IIter, _IIter, _OIter, _BinaryPredicate)
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>
  constexpr _OutputIterator unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)
- ios_base & unitbuf (ios_base &__base)
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >

- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< _Key, _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>

unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >

- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type={}, ↩ _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< _Key, _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = ↩ _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_Input↩ Iterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (initializer_list< _Tp >, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Allocator) -> unordered↩ _multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _↩ RequireAllocator<_Allocator>>
  unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_↩ Allocator>>
  unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _↩ Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, _Allocator) -> unordered_set< typename iterator_traits< _Input↩ Iterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>

unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >

- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< _Tp, _Hash, _Pred, _Allocator >
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr _FIter **upper_bound** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type, typename _Compare>
  constexpr _FIter **upper_bound** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- ios_base & uppercase (ios_base &__base)
- template<typename _Facet>
  const _Facet & use_facet (const locale &__loc)
- template const codecvt< char, char, mbstate_t > & **use_facet**< **codecvt**< **char, char, mbstate_t** > > (const locale &)
- template const codecvt< wchar_t, char, mbstate_t > & **use_facet**< **codecvt**< **wchar_t, char, mbstate_t** > > (const locale &)
- template<_Std_pair _Tp, typename _Alloc>
  constexpr auto **uses_allocator_construction_args** (const _Alloc &) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **uses_allocator_construction_args** (const _Alloc &, _Up &&, _Vp &&) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **uses_allocator_construction_args** (const _Alloc &, const pair< _Up, _Vp > &&) noexcept

- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **uses_allocator_construction_args** (const _Alloc &, const pair< _Up, _Vp > &) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **uses_allocator_construction_args** (const _Alloc &, pair< _Up, _Vp > &&) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **uses_allocator_construction_args** (const _Alloc &, pair< _Up, _Vp > &) noexcept
- template<typename _Tp, typename _Alloc, typename... _Args>
  requires (! _Std_pair<_Tp>)
  constexpr auto **uses_allocator_construction_args** (const _Alloc &__a, _Args &&... __args) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Tuple1, typename _Tuple2>
  constexpr auto **uses_allocator_construction_args** (const _Alloc &__a, piecewise_construct_t, _Tuple1 &&__x, _Tuple2 &&__y) noexcept
- template<typename _Tp, size_t _Nm>
  valarray (const _Tp(&)[_Nm], size_t) -> valarray< _Tp >
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  vector (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> vector< _ValT, _Allocator >
- wchar_t * **wcschr** (wchar_t *__p, wchar_t __c)
- wchar_t * **wcspbrk** (wchar_t *__s1, const wchar_t *__s2)
- wchar_t * **wcsrchr** (wchar_t *__p, wchar_t __c)
- wchar_t * **wcsstr** (wchar_t *__s1, const wchar_t *__s2)
- template<typename _Tp>
  weak_ptr (shared_ptr< _Tp >) -> weak_ptr< _Tp >
- wchar_t * **wmemchr** (wchar_t *__p, wchar_t __c, size_t __n)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & ws (basic_istream< _CharT, _Traits > &__is)

- template<size_t _Nb>
  constexpr bitset< _Nb > operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  constexpr bitset< _Nb > operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  constexpr bitset< _Nb > operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept

- template<class _CharT, class _Traits, size_t _Nb>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)
- template<class _CharT, class _Traits, size_t _Nb>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)

- template<typename _Tp>
  constexpr complex< _Tp > operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator+ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator+ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator- (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator- (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator* (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator* (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator/ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > operator/ (const _Tp &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr bool operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr bool operator== (const complex< _Tp > &__x, const _Tp &__y)

- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)
- template<class _Traits>
  basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)
- template<class _Traits>
  basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char &__c)

- template<typename _CharT, typename _Traits, size_t _Num>
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__in, _CharT(&__s)[_↩
  Num])
- template<class _Traits, size_t _Num>
  basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, unsigned char(&__s)[_↩
  Num])
- template<class _Traits, size_t _Num>
  basic_istream< char, _Traits > & operator>> (basic_istream< char, _Traits > &__in, signed char(&__s)[_↩
  Num])

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, char __c)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, signed char __c)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const char *__s)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)
- template<typename _Traits>
  basic_ostream< char, _Traits > & operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)

Matching, Searching, and Replacing

- template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
  bool regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type, typename _Alloc, typename _Rx_traits>
  bool regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch↩ _type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete
- template<typename _Ch_type, class _Rx_traits>
  bool regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>
  bool regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
  bool regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)

- template<typename _Ch_type, class _Alloc, class _Rx_traits>
  bool regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_type, typename _Rx_traits>
  bool regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>
  bool regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch↩
  _type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  _Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, ↩
  _Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
  _Out_iter regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _↩
  Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa>
  basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > &__↩
  s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  basic_string< _Ch_type, _St, _Sa > regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __↩
  flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  basic_string< _Ch_type > regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_↩
  traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↩
  flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type>
  basic_string< _Ch_type > regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)

- template<typename _Tp, _Lock_policy _Lp>
  bool atomic_is_lock_free (const __shared_ptr< _Tp, _Lp > *)
- template<typename _Tp>
  bool atomic_is_lock_free (const shared_ptr< _Tp > *__p)

- template<typename _Tp>
  shared_ptr< _Tp > atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order)
- template<typename _Tp>
  shared_ptr< _Tp > atomic_load (const shared_ptr< _Tp > *__p)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > atomic_load_explicit (const __shared_ptr< _Tp, _Lp > *__p, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > atomic_load (const __shared_ptr< _Tp, _Lp > *__p)

- template<typename _Tp>
  void atomic_store_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)
- template<typename _Tp>
  void atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp, _Lock_policy _Lp>
  void atomic_store_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  void atomic_store (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)

- template<typename _Tp>
  shared_ptr< _Tp > atomic_exchange_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)
- template<typename _Tp>
  shared_ptr< _Tp > atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > atomic_exchange_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > atomic_exchange (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)

- template<typename _Tp>
  bool atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order, memory_order)
- template<typename _Tp>
  bool atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp>
  bool atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  bool atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp, _Lock_policy _Lp>
  bool atomic_compare_exchange_strong_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  bool atomic_compare_exchange_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)
- template<typename _Tp, _Lock_policy _Lp>
  bool atomic_compare_exchange_weak_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp, _Lock_policy _Lp>
  bool atomic_compare_exchange_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, _shared_ptr< _Tp, _Lp > __w)

- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & get (pair< _Tp1, _Tp2 > &__in) noexcept

- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && get (pair< _Tp1, _Tp2 > &&__in) noexcept
- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & get (const pair< _Tp1, _Tp2 > &__in) noexcept
- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && get (const pair< _Tp1, _Tp2 > &&__in) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp & get (pair< _Tp, _Up > &__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp & get (const pair< _Tp, _Up > &__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp && get (pair< _Tp, _Up > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp && get (const pair< _Tp, _Up > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp & get (pair< _Up, _Tp > &__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp & get (const pair< _Up, _Tp > &__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp && get (pair< _Up, _Tp > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp && get (const pair< _Up, _Tp > &&__p) noexcept

Variables

- template<typename _ValT, typename _Tp>
  constexpr bool **__can_use_memchr_for_find**
- const _Lock_policy **__default_lock_policy**
- template<typename _Tp>
  constexpr auto **__denorm_min_v**
- template<typename _Tp>
  constexpr auto **__digits10_v**
- template<typename _Tp>
  constexpr auto **__digits_v**
- template<typename _Tp>
  constexpr auto **__epsilon_v**
- template<typename _Tp>
  constexpr auto **__finite_max_v**
- template<typename _Tp>
  constexpr auto **__finite_min_v**
- template<template< typename > class _Trait, typename _Tp>
  constexpr bool **__has_iec559_behavior_v**
- template<typename _Tp>
  constexpr bool **__has_iec559_storage_format_v**
- template<typename _Tp>
  constexpr auto **__infinity_v**
- static ios_base::Init **__ioinit**
- template<typename>
  constexpr bool **__is_constant_wrapper_v**
- template<auto __cw, typename _Fn>
  constexpr bool **__is_constant_wrapper_v**< **constant_wrapper**< **__cw, _Fn** > >

- template<typename _Tp, typename = void>
  constexpr bool **__is_hash_enabled_for**
- template<typename _Tp>
  constexpr bool **__is_hash_enabled_for**< **_Tp, __void_t**< **decltype(hash**< **_Tp** >()(declval< **_Tp** >()))> >
- template<typename>
  constexpr bool **__is_in_place_index_v**
- template<size_t _Nm>
  constexpr bool **__is_in_place_index_v**< **in_place_index_t**< **_Nm** > >
- template<typename>
  constexpr bool **__is_in_place_type_v**
- template<typename _Tp>
  constexpr bool **__is_in_place_type_v**< **in_place_type_t**< **_Tp** > >
- template<typename _Tp, typename... _Args>
  constexpr bool **__is_nothrow_new_constructible**
- template<typename _Tp>
  constexpr bool **__is_pair**
- template<typename _Tp, typename _Up>
  constexpr bool **__is_pair**< **pair**< **_Tp, _Up** > >
- template<typename _Tp>
  const bool **__is_rand_dist_float_v**
- template<template< typename > class>
  constexpr bool **__is_std_op_template**
- template<> constexpr bool **__is_std_op_template**< **std::bit_and** >
- template<> constexpr bool **__is_std_op_template**< **std::bit_not** >
- template<> constexpr bool **__is_std_op_template**< **std::bit_or** >
- template<> constexpr bool **__is_std_op_template**< **std::bit_xor** >
- template<> constexpr bool **__is_std_op_template**< **std::divides** >
- template<> constexpr bool **__is_std_op_template**< **std::equal_to** >
- template<> constexpr bool **__is_std_op_template**< **std::greater** >
- template<> constexpr bool **__is_std_op_template**< **std::greater_equal** >
- template<> constexpr bool **__is_std_op_template**< **std::less** >
- template<> constexpr bool **__is_std_op_template**< **std::less_equal** >
- template<> constexpr bool **__is_std_op_template**< **std::logical_and** >
- template<> constexpr bool **__is_std_op_template**< **std::logical_not** >
- template<> constexpr bool **__is_std_op_template**< **std::logical_or** >
- template<> constexpr bool **__is_std_op_template**< **std::minus** >
- template<> constexpr bool **__is_std_op_template**< **std::modulus** >
- template<> constexpr bool **__is_std_op_template**< **std::multiplies** >
- template<> constexpr bool **__is_std_op_template**< **std::negate** >
- template<> constexpr bool **__is_std_op_template**< **std::not_equal_to** >
- template<> constexpr bool **__is_std_op_template**< **std::plus** >
- template<typename _Fn>
  constexpr bool **__is_std_op_wrapper**
- template<template< typename > class _Ft, typename _Tp>
  constexpr bool **__is_std_op_wrapper**< **_Ft**< **_Tp** > >
- template<typename _Tp>
  constexpr bool **__is_submdspan_mapping_result**
- template<typename _Mapping>
  constexpr bool **__is_submdspan_mapping_result**< **submdspan_mapping_result**< **_Mapping** > >
- template<typename _It, typename _Sent, ranges::subrange_kind _Kind>
  constexpr bool **__is_tuple_like_v**< **ranges::subrange**< **_It, _Sent, _Kind** > >

- template<typename _Tp>
  constexpr auto **__max_digits10_v**
- template<typename _Tp>
  constexpr auto **__max_exponent10_v**
- template<typename _Tp>
  constexpr auto **__max_exponent_v**
- template<typename _Tp>
  constexpr auto **__min_exponent10_v**
- template<typename _Tp>
  constexpr auto **__min_exponent_v**
- template<typename _Tp>
  constexpr auto **__norm_min_v**
- template<typename _Tp>
  constexpr auto **__quiet_NaN_v**
- template<typename _Tp>
  constexpr auto **__radix_v**
- template<typename _Tp>
  constexpr auto **__reciprocal_overflow_threshold_v**
- template<typename _Tp>
  constexpr auto **__round_error_v**
- template<typename _Tp>
  constexpr auto **__signaling_NaN_v**
- template<typename _CharT>
  locale::id **__timepunct**< **_CharT** >**::id**
- template<typename _CharT>
  const _CharT * **__timepunct_cache**< **_CharT** >**::_S_timezones** [14]
- template<> const char * **__timepunct_cache**< **char** >**::_S_timezones** [14]
- template<> const wchar_t * **__timepunct_cache**< **wchar_t** >**::_S_timezones** [14]
- template<template< typename > class _Trait, typename _Tp>
  constexpr bool **__value_exists_v**
- constexpr adopt_lock_t adopt_lock
- template<size_t _Len, typename... _Types>
  const size_t **aligned_union**< **_Len, _Types...** >**::alignment_value**
- template<typename _CharT, typename _Traits, typename _Alloc>
  const basic_string< _CharT, _Traits, _Alloc >::size_type **basic_string**< **_CharT, _Traits, _Alloc** >**::npos**
- template<typename _InternT, typename _ExternT, typename _StateT>
  locale::id **codecvt**< **_InternT, _ExternT, _StateT** >**::id**
- template<typename _InternT, typename _ExternT>
  locale::id **codecvt**< **_InternT, _ExternT, encoding_state** >**::id**
- template<typename _CharT>
  locale::id **collate**< **_CharT** >**::id**
- constexpr __compare::_Partial_fallback **compare_partial_order_fallback**
- constexpr __compare::_Strong_fallback **compare_strong_order_fallback**
- constexpr __compare::_Weak_fallback **compare_weak_order_fallback**
- template<typename _CharT>
  locale::id **ctype**< **_CharT** >**::id**
- template<auto _Xv>
  constexpr auto **cw**
- constexpr default_sentinel_t default_sentinel
- constexpr defer_lock_t defer_lock
- constexpr destroying_delete_t destroying_delete

- template<typename _Sent, typename _Iter>
  constexpr bool **disable_sized_sentinel_for**
- template<typename _Iterator1, typename _Iterator2>
  constexpr bool **disable_sized_sentinel_for**< **move_iterator**< **_Iterator1** >, **move_iterator**< **_Iterator2** > >
- template<typename _Iterator1, typename _Iterator2>
  constexpr bool **disable_sized_sentinel_for**< **reverse_iterator**< **_Iterator1** >, **reverse_iterator**< **_Iterator2** > >
- constexpr full_extent_t **full_extent**
- constexpr _Swallow_assign ignore
- constexpr in_place_t **in_place**
- template<size_t _Idx>
  constexpr in_place_index_t< _Idx > **in_place_index**
- template<typename _Tp>
  constexpr in_place_type_t< _Tp > **in_place_type**
- template<typename _Tp>
  constexpr bool **is_bind_expression_v**
- template<typename _Tp>
  constexpr bool **is_error_code_enum_v**
- template<typename _Tp>
  constexpr bool **is_error_condition_enum_v**
- template<typename _Tp>
  constexpr int **is_placeholder_v**
- constexpr memory_order **memory_order_acq_rel**
- constexpr memory_order **memory_order_acquire**
- constexpr memory_order **memory_order_consume**
- constexpr memory_order **memory_order_relaxed**
- constexpr memory_order **memory_order_release**
- constexpr memory_order **memory_order_seq_cst**
- template<typename _CharT>
  locale::id **messages**< **_CharT** >**::id**
- template<typename _CharT, typename _InIter>
  locale::id **money_get**< **_CharT, _InIter** >**::id**
- template<typename _CharT, typename _OutIter>
  locale::id **money_put**< **_CharT, _OutIter** >**::id**
- template<typename _CharT, bool _Intl>
  locale::id **moneypunct**< **_CharT, _Intl** >**::id**
- template<typename _CharT, bool _Intl>
  const bool **moneypunct**< **_CharT, _Intl** >**::intl**
- template<typename _CharT, bool _Intl>
  const bool **moneypunct_byname**< **_CharT, _Intl** >**::intl**
- const nothrow_t **nothrow**
- template<typename _CharT, typename _InIter>
  locale::id **num_get**< **_CharT, _InIter** >**::id**
- template<typename _CharT, typename _OutIter>
  locale::id **num_put**< **_CharT, _OutIter** >**::id**
- template<typename _CharT>
  locale::id **numpunct**< **_CharT** >**::id**
- constexpr __compare::_Partial_order **partial_order**
- constexpr piecewise_construct_t piecewise_construct
- template<typename _R1, typename _R2>
  constexpr bool **ratio_equal_v**

- template<typename _R1, typename _R2>
  constexpr bool **ratio_greater_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_greater_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_less_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_less_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_not_equal_v**
- constexpr sorted_equivalent_t **sorted_equivalent**
- constexpr sorted_unique_t **sorted_unique**
- constexpr __compare::_Strong_order **strong_order**
- template<typename _CharT, typename _InIter>
  locale::id **time_get< _CharT, _InIter >::id**
- template<typename _CharT, typename _OutIter>
  locale::id **time_put< _CharT, _OutIter >::id**
- constexpr try_to_lock_t try_to_lock
- template<typename _Tp>
  constexpr size_t **tuple_size_v**
- template<typename _Tp, size_t _Nm>
  constexpr size_t **tuple_size_v< array< _Tp, _Nm > >**
- template<typename _Tp, size_t _Nm>
  constexpr size_t **tuple_size_v< const array< _Tp, _Nm > >**
- template<typename _Tp1, typename _Tp2>
  constexpr size_t **tuple_size_v< const pair< _Tp1, _Tp2 > >**
- template<typename... _Types>
  constexpr size_t **tuple_size_v< const tuple< _Types... > >**
- template<typename _Tp1, typename _Tp2>
  constexpr size_t **tuple_size_v< pair< _Tp1, _Tp2 > >**
- template<typename... _Types>
  constexpr size_t **tuple_size_v< tuple< _Types... > >**
- template<typename _Tp, typename _Up>
  constexpr strong_ordering type_order_v
- constexpr unreachable_sentinel_t **unreachable_sentinel**
- constexpr __compare::_Weak_order **weak_order**

Standard Stream Objects

The <iostream> header declares the eight standard stream objects. For other declarations, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html and the I/O forward declarations

They are required by default to cooperate with the global C library's FILE streams, and to be available during program startup and termination. For more information, see the section of the manual linked to above.

- istream cin
- ostream cout
- ostream cerr
- ostream clog
- wistream wcin
- wostream wcout
- wostream wcerr
- wostream wclog

- template<typename... _Tp>
  using common_reference_t

- template<typename _Tp, typename _Up>
  constexpr bool is_layout_compatible_v
- template<typename _Base, typename _Derived>
  constexpr bool is_pointer_interconvertible_base_of_v
- template<typename _S1, typename _S2, typename _M1, typename _M2>
  constexpr bool is_corresponding_member (_M1 _S1::*__m1, _M2 _S2::*__m2) noexcept
- template<typename _Tp, typename _Mem>
  constexpr bool is_pointer_interconvertible_with_class (_Mem _Tp::*__mp) noexcept

4.10.1 Detailed Description

ISO C++ entities toplevel namespace is std.

4.10.2 Typedef Documentation

__ptr_rebind

```
template<typename _Ptr, typename _Tp>
using std::__ptr_rebind
```

Convenience alias for rebinding pointers.

__umap_traits

```
template<bool _Cache>
using std::__umap_traits
```

Base types for unordered_map.

__ummap_traits

```
template<bool _Cache>
using std::__ummap_traits
```

Base types for unordered_multimap.

__umset_traits

```
template<bool _Cache>
using std::__umset_traits
```

Base types for unordered_multiset.

__uset_traits

```
template<bool _Cache>
using std::__uset_traits
```

Base types for unordered_set.

compare_three_way_result_t

```
template<typename _Tp, typename _Up = _Tp>
using std::compare_three_way_result_t
```

[cmp.result], result of three-way comparison

index_sequence

```
template<size_t... _Idx>
using std::index_sequence
```

Alias template index_sequence.

index_sequence_for

```
template<typename... _Types>
using std::index_sequence_for
```
Alias template index_sequence_for.

iter_rvalue_reference_t

```
template<__detail::__dereferenceable _Tp>
using std::iter_rvalue_reference_t
```
The result type of ranges::iter_move(std::declval<_Tp&>()).

make_index_sequence

```
template<size_t _Num>
using std::make_index_sequence
```
Alias template make_index_sequence.

make_integer_sequence

```
template<typename _Tp, _Tp _Num>
using std::make_integer_sequence
```
Alias template make_integer_sequence.

new_handler

```
typedef void(* std::new_handler) ()
```
If you write your own error handler to be called by `new`, it must be of this type.

projected

```
template<indirectly_readable _Iter, indirectly_regular_unary_invocable< _Iter > _Proj>
using std::projected
```
[projected], projected

streamoff

```
typedef long long std::streamoff
```
Type used by fpos, char_traits<char>, and char_traits<wchar_t>.
In clauses 21.1.3.1 and 27.4.1 streamoff is described as an implementation defined type. Note: In versions of GCC up to and including GCC 3.3, streamoff was typedef long.

streampos

```
typedef fpos<mbstate_t> std::streampos
```
File position for char streams.

streamsize

```
typedef ptrdiff_t std::streamsize
```
Integral type for I/O operation counts and buffer sizes.

u16streampos

```
typedef fpos<mbstate_t> std::u16streampos
```
File position for char16_t streams.

u32streampos

typedef fpos<mbstate_t> std::u32streampos
File position for char32_t streams.

wstreampos

typedef fpos<mbstate_t> std::wstreampos
File position for wchar_t streams.

4.10.3 Enumeration Type Documentation

chars_format

enum class std::chars_format [strong]
floating-point format for primitive numerical conversion

float_denorm_style

enum std::float_denorm_style
Describes the denormalization for floating-point types.
These values represent the presence or absence of a variable number of exponent bits. This type is used in the std::↩ numeric_limits class.

Enumerator

denorm_indeterminate	Indeterminate at compile time whether denormalized values are allowed.
denorm_absent	The type does not allow denormalized values.
denorm_present	The type allows denormalized values.

float_round_style

enum std::float_round_style
Describes the rounding style for floating-point types.
This is used in the std::numeric_limits class.

Enumerator

round_indeterminate	Intermediate.
round_toward_zero	To zero.
round_to_nearest	To the nearest representable value.
round_toward_infinity	To infinity.
round_toward_neg_infinity	To negative infinity.

io_errc

enum class std::io_errc [strong]
I/O error code.

4.10.4 Function Documentation

__gen_two_uniform_ints()

```
template<typename _IntType, typename _UniformRandomBitGenerator>
pair< _IntType, _IntType > std::__gen_two_uniform_ints (
            _IntType __b0,
            _IntType __b1,
            _UniformRandomBitGenerator && __g)
```

Generate two uniformly distributed integers using a single distribution invocation.

Parameters

__b0	The upper bound for the first integer.
__b1	The upper bound for the second integer.
__g	A UniformRandomBitGenerator.

Returns

A pair (i, j) with i and j uniformly distributed over [0, __b0) and [0, __b1), respectively.

Requires: __b0 * __b1 <= __g.max() - __g.min().
Using uniform_int_distribution with a range that is very small relative to the range of the generator ends up wasting potentially expensively generated randomness, since uniform_int_distribution does not store leftover randomness between invocations.
If we know we want two integers in ranges that are sufficiently small, we can compose the ranges, use a single distribution invocation, and significantly reduce the waste.
References make_pair().
Referenced by shuffle().

__lg()

```
template<typename _Tp>
_Tp std::__lg (
            _Tp __n)  [inline], [constexpr]
```

This is a helper function for the sort routines and for random.tcc.
Referenced by nth_element(), nth_element(), std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator()(), and std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed().

_Construct()

```
template<typename _Tp, typename...  _Args>
void std::_Construct (
            _Tp * __p,
            _Args &&...  __args)  [inline], [constexpr]
```

Constructs an object in existing memory by invoking an allocated object's constructor with an initializer.
References forward().
Referenced by std::allocator_traits< _OuterAlloc >::construct(), std::allocator_traits< allocator< _Tp > >::construct(), and std::allocator_traits< allocator< void > >::construct().

_Destroy() [1/2]

```
template<typename _ForwardIterator>
void std::_Destroy (
```

```
            _ForwardIterator __first,
            _ForwardIterator __last)  [inline], [constexpr]
```

Destroy a range of objects. If the value_type of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.
References _Destroy(), and addressof().
Referenced by std::vector< _Tp, polymorphic_allocator< _Tp > >::~vector(), _Destroy(), _Destroy_n(), std::deque< _Tp, _Alloc >::_M_
std::deque< _Tp, _Alloc >::_M_range_initialize(), std::allocator_traits< _OuterAlloc >::destroy(), std::allocator_traits< allocator< void >
std::vector< _Tp, _Alloc >::operator=(), and std::vector< _Tp, _Alloc >::reserve().

_Destroy() [2/2]

```
template<typename _Tp>
void std::_Destroy (
            _Tp * __pointer)  [inline], [constexpr]
```

Destroy the object pointed to by a pointer type.

_Destroy_n()

```
template<typename _ForwardIterator, typename _Size>
_ForwardIterator std::_Destroy_n (
            _ForwardIterator __first,
            _Size __count)  [inline], [constexpr]
```

Destroy a range of objects. If the value_type of the object has a trivial destructor, the compiler should optimize all of this away, otherwise the objects' destructors must be invoked.
References _Destroy(), addressof(), and advance().

abs()

```
template<typename _Tp>
_Tp std::abs (
            const complex< _Tp > & __z)  [inline]
```

Return magnitude of *z*.
References abs(), and arg().

acosh()

```
float std::acosh (
            float __x)  [constexpr]
```

Additional overloads.

advance()

```
template<typename _InputIterator, typename _Distance>
void std::advance (
            _InputIterator & __i,
            _Distance __n)  [inline], [constexpr]
```

A generalization of pointer arithmetic.

Parameters

_↩ _*i*	An input iterator.
_↩ _*n*	The *delta* by which to change __i.

This increments i by n. For bidirectional and random access iterators, __n may be negative, in which case __i is decremented.
For random access iterators, this uses their + and − operations and are constant time. For other iterator classes they are linear time.
Referenced by _Destroy_n(), std::deque< _Tp, _Alloc >::_M_range_initialize(), __gnu_pbds::detail::pat_trie_base::_Node_citer< node, l __gnu_pbds::detail::pat_trie_base::_Node_iter< node, leaf, head, inode, const_iterator, iterator, _Alloc >::get_child(), and partition_point().

begin() [1/3]

```
template<typename _Container>
auto std::begin (
            _Container & __cont) -> decltype(__cont.begin())  [inline], [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the container.

Parameters

__*cont*	Container.

Since

C++11

Referenced by cbegin(), std::vector< _Tp, _Alloc >::insert(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::remove(), std::list< _Tp, _Alloc >::remove_if(), std::list< _Tp, _Alloc >::sort(), std::list< _Tp, _Alloc >::sort(), std::list< _Tp, _Alloc >::unique(), and std::list< _Tp, _Alloc >::unique().

begin() [2/3]

```
template<typename _Tp, size_t _Nm>
_Tp * std::begin (
            _Tp(&) __arr[_Nm])  [inline], [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the array.

Parameters

__*arr*	Array.

Since

C++11

begin() [3/3]

```
template<typename _Container>
auto std::begin (
            const _Container & __cont) -> decltype(__cont.begin())  [inline], [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the const container.

Parameters

__cont	Container.

boolalpha()

```
ios_base & std::boolalpha (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::boolalpha).
References std::ios_base::boolalpha.

cbegin()

```
template<typename _Container>
auto std::cbegin (
            const _Container & __cont) -> decltype(std::begin(__cont))  [constexpr], [noexcept]
```

Return an iterator pointing to the first element of the const container.

Parameters

__cont	Container.

Since

C++14

References begin().
Referenced by std::vector< _Tp, _Alloc >::insert().

cend()

```
template<typename _Container>
auto std::cend (
            const _Container & __cont) -> decltype(std::end(__cont))  [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the const container.

Parameters

__cont	Container.

Since

C++14

References end().

const_pointer_cast()

```
template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::const_pointer_cast (
            const __shared_ptr< _Tp1, _Lp > & __r)  [inline], [noexcept]
```

const_pointer_cast

crbegin()

```
template<typename _Container>
auto std::crbegin (
            const _Container & __cont) -> decltype(std::rbegin(__cont))  [inline], [constexpr],
[noexcept]
```

Return a reverse iterator pointing to the last element of the const container.

Parameters

__cont	Container.

Since

C++14

References rbegin().

crend()

```
template<typename _Container>
auto std::crend (
            const _Container & __cont) -> decltype(std::rend(__cont))  [inline], [constexpr],
[noexcept]
```

Return a reverse iterator pointing one past the first element of the const container.

Parameters

__cont	Container.

Since

C++14

References rend().

data() [1/3]

```
template<typename _Container>
auto std::data (
            _Container & __cont) -> decltype(__cont.data())  [nodiscard], [constexpr], [noexcept]
```

Return the data pointer of a container.

Parameters

__cont	Container.

Since

C++17

Referenced by std::basic_stringbuf< char >::basic_stringbuf(), and std::basic_string< _CharT, _Traits, _Alloc >::find().

data() [2/3]

```
template<typename _Tp, size_t _Nm>
_Tp * std::data (
            _Tp(&) __array[_Nm])  [nodiscard], [constexpr], [noexcept]
```

Return the data pointer of an array.

Parameters

__array	Array.

Since

C++17

data() [3/3]

```
template<typename _Container>
auto std::data (
            const _Container & __cont) -> decltype(__cont.data())  [nodiscard], [constexpr],
[noexcept]
```

Return the data pointer of a const container.

Parameters

__cont	Container.

Since

C++17

dec()

```
ios_base & std::dec (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::dec, ios_base::basefield).
References std::ios_base::basefield, and std::ios_base::dec.

defaultfloat()

```
ios_base & std::defaultfloat (
            ios_base & __base)  [inline]
```

Calls `base.unsetf(ios_base::floatfield)`.
References std::ios_base::floatfield.

distance()

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

A generalization of pointer arithmetic.

Parameters

__first,__last	Input iterators that form a valid range.

Returns

The distance between them.

Returns `n` such that `__first + n == __last`. This requires that `__last` must be reachable from `__first`, or for random access iterators either `__last` is reachable from `__first` or `__first` is reachable from `__last`. In the latter case, `n` may be negative.
For random access iterators, this uses their + and − operations and is constant time. For other iterator classes they are linear time.
References move().
Referenced by std::deque< _Tp, _Alloc >::_M_range_initialize(), is_heap(), is_heap_until(), is_heap_until(), std::sub_match< const char * >::length(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_pbds::detail::pat_trie_base::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >::num↩_children(), partition_point(), std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _↩BidirectionalIterator > > >< const char * >::position(), __gnu_cxx::random_sample_n(), __gnu_cxx::random_sample_n(), std::list< _Tp, polymorphic_allocator< _Tp > >::size(), and std::list< _Tp, polymorphic_allocator< _Tp > >::splice().

dynamic_pointer_cast()

```
template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::dynamic_pointer_cast (
            const __shared_ptr< _Tp1, _Lp > & __r)  [inline], [noexcept]
```

dynamic_pointer_cast

empty() [1/2]

```
template<typename _Container>
auto std::empty (
            const _Container & __cont) -> decltype(__cont.empty())  [nodiscard], [constexpr],
[noexcept]
```

Return whether a container is empty.

Parameters

__cont	Container.

Since

C++17

Referenced by std::list< _Tp, _Alloc >::sort(), and std::list< _Tp, _Alloc >::sort().

empty() [2/2]

```
template<typename _Tp, size_t _Nm>
bool std::empty (
            const _Tp(&)[_Nm]) [nodiscard], [constexpr], [noexcept]
```

Return whether an array is empty (always false).

Parameters

-	An array.

Since

C++17

end() [1/3]

```
template<typename _Container>
auto std::end (
            _Container & __cont) -> decltype(__cont.end()) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the container.

Parameters

__cont	Container.

Since

C++11

Referenced by cend(), std::vector< _Tp, _Alloc >::insert(), std::list< _Tp, _Alloc >::merge(), std::list< _Tp, _Alloc >::merge(), std::vector< _Tp, _Alloc >::operator=(), std::list< _Tp, _Alloc >::remove(), std::list< _Tp, _Alloc >::remove_if(), std::list< _Tp, _Alloc >::resize(), std::list< _Tp, _Alloc >::resize(), std::list< _Tp, _Alloc >::sort(), std::list< _Tp, _Alloc >::sort(), std::list< _Tp, _Alloc >::unique(), and std::list< _Tp, _Alloc >::unique().

end() [2/3]

```
template<typename _Tp, size_t _Nm>
_Tp * std::end (
            _Tp(&) __arr[_Nm]) [inline], [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the array.

Parameters

__arr	Array.

Since

C++11

end() [3/3]

```
template<typename _Container>
auto std::end (
            const _Container & __cont) -> decltype(__cont.end())  [inline], [constexpr], [noexcept]
```

Return an iterator pointing to one past the last element of the const container.

Parameters

__cont	Container.

Since

C++11

endl()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::endl (
            basic_ostream< _CharT, _Traits > & __os)  [inline]
```

Write a newline and flush the stream.

This manipulator is often mistakenly used when a simple newline is desired, leading to poor buffering performance. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.↩streambuf.buffering for more on this subject.

References flush(), std::basic_ostream< _CharT, _Traits >::put(), and std::basic_ios< _CharT, _Traits >::widen().

ends()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::ends (
            basic_ostream< _CharT, _Traits > & __os)  [inline]
```

Write a null character into the output sequence.

Null character is `CharT()` by definition. For CharT of `char`, this correctly writes the ASCII `NUL` character string terminator.

References std::basic_ostream< _CharT, _Traits >::put().

fabs()

```
template<typename _Tp>
_Tp std::fabs (
            const std::complex< _Tp > & __z)  [inline]
```

fabs(__z) TR1 8.1.8 [tr.c99.cmplx.fabs]

References abs(), and fabs().

Referenced by fabs().

fixed()

```
ios_base & std::fixed (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::fixed, ios_base::floatfield).

References std::ios_base::fixed, and std::ios_base::floatfield.

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::flush (
            basic_ostream< _CharT, _Traits > & __os)  [inline]
```

Flushes the output stream.
This manipulator simply calls the stream's `flush()` member function.
References std::basic_ostream< _CharT, _Traits >::flush().
Referenced by endl().

from_chars()

```
template<typename _Tp, enable_if_t< __or_< __is_signed_or_unsigned_integer< _Tp >, is_same<
char, _Tp > >::value, int > = 0>
from_chars_result std::from_chars (
            const char * __first,
            const char * __last,
            _Tp & __value,
            int __base = 10)  [constexpr]
```

std::from_chars for integer types.
References std::__detail::__from_chars_alnum(), and std::__detail::__from_chars_pow2_base().

generate_canonical()

```
template<typename _RealT, size_t __digits, typename _Urbg>
_RealT std::generate_canonical (
            _Urbg & __urng)
```

Produce a random floating-point value in the range [0..1)
The result of `std::generate_canonical<RealT,digits>(urng)` is a random floating-point value of type `RealT` in the range [0..1), using entropy provided by the uniform random bit generator `urng`. A value for `digits` may be passed to limit the precision of the result to so many bits, but normally `-1u` is passed to get the native precision of `RealT`. As many `urng()` calls are made as needed to obtain the required entropy. On rare occasions, more `urng()` calls are used. It is fastest when the value of `Urbg::max()` is a power of two less one, such as from a `std::philox4x32` (for `float`) or `philox4x64` (for `double`).

Since

C++11

References std::__numeric_limits_base::digits, generate_canonical(), and std::__numeric_limits_base::radix.

get() [1/12]

```
template<typename _Tp, typename _Up>
const _Tp && std::get (
            const pair< _Tp, _Up > && __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair
References forward().

get() [2/12]

```
template<typename _Tp, typename _Up>
const _Tp & std::get (
            const pair< _Tp, _Up > & __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

get() [3/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (
            const pair< _Tp1, _Tp2 > && __in)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair
References move().

get() [4/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (
            const pair< _Tp1, _Tp2 > & __in)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

get() [5/12]

```
template<typename _Tp, typename _Up>
const _Tp && std::get (
            const pair< _Up, _Tp > && __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair
References forward().

get() [6/12]

```
template<typename _Tp, typename _Up>
const _Tp & std::get (
            const pair< _Up, _Tp > & __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

get() [7/12]

```
template<typename _Tp, typename _Up>
_Tp && std::get (
            pair< _Tp, _Up > && __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair
References forward().

get() [8/12]

```
template<typename _Tp, typename _Up>
_Tp & std::get (
            pair< _Tp, _Up > & __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

get() [9/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (
            pair< _Tp1, _Tp2 > && __in)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair
References move().

get() [10/12]

```
template<size_t _Int, class _Tp1, class _Tp2>
tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (
            pair< _Tp1, _Tp2 > & __in)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

get() [11/12]

```
template<typename _Tp, typename _Up>
_Tp && std::get (
            pair< _Up, _Tp > && __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair
References forward().

get() [12/12]

```
template<typename _Tp, typename _Up>
_Tp & std::get (
            pair< _Up, _Tp > & __p)  [constexpr], [noexcept]
```

std::get overloads for accessing members of std::pair

get_money()

```
template<typename _MoneyT>
_Get_money< _MoneyT > std::get_money (
            _MoneyT & __mon,
            bool __intl = false)  [inline]
```

Extended manipulator for extracting money.

Parameters

__mon	Either long double or a specialization of `basic_string`.
__intl	A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator extracts *__mon*.

get_new_handler()

```
new_handler std::get_new_handler ()  [noexcept]
```

Return the current new handler.

get_temporary_buffer()

```
template<typename _Tp>
pair< _Tp *, ptrdiff_t > std::get_temporary_buffer (
            ptrdiff_t __len)  [noexcept]
```

Allocates a temporary buffer.

Parameters

__len	The number of objects of type Tp.

Returns

See full description.

Reinventing the wheel, but this time with prettier spokes!
This function tries to obtain storage for __len adjacent Tp objects. The objects themselves are not constructed, of course. A pair<> is returned containing *the buffer's address and capacity (in the units of sizeof(_Tp)), or a pair of 0 values if no storage can be obtained.* Note that the capacity obtained may be less than that requested if the memory is unavailable; you should compare len with the .second return value.
Provides the nothrow exception guarantee.

get_time()

```
template<typename _CharT>
_Get_time< _CharT > std::get_time (
            std::tm * __tmb,
            const _CharT * __fmt)  [inline]
```

Extended manipulator for extracting time.
This manipulator uses time_get::get to extract time. [ext.manip]

Parameters

__tmb	struct to extract the time data to.
__fmt	format string.

getline() [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
            basic_istream< _CharT, _Traits > && __is,
            basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Read a line from an rvalue stream into a string.
References getline().

getline() [2/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
            basic_istream< _CharT, _Traits > && __is,
            basic_string< _CharT, _Traits, _Alloc > & __str,
            _CharT __delim)  [inline]
```

Read a line from an rvalue stream into a string.
References getline().

getline() [3/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_istream< _CharT, _Traits > & std::getline (
            basic_istream< _CharT, _Traits > & __is,
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Read a line from stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from is into *__str* until '
' is found, the end of the stream is encountered, or str.max_size() is reached. If is.width() is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased. If end of line was encountered, it is extracted but not stored into *__str*.
References getline(), and std::basic_ios< _CharT, _Traits >::widen().

getline() [4/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_istream< _CharT, _Traits > & std::getline (
             basic_istream< _CharT, _Traits > & __is,
             __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
             _CharT __delim)
```

Read a line from stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.
__delim	Character marking end of line.

Returns

Reference to the input stream.

Stores characters from *__is* into *__str* until *__delim* is found, the end of the stream is encountered, or str.max_size() is reached. If is.width() is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased. If *delim* was encountered, it is extracted but not stored into *__str*.
References __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append(), __gnu_cxx::__versa_string< _CharT, _Traits, _Allo
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size(), std::basic_ios< _CharT, _Traits >::rdbuf(),
and std::basic_ios< _CharT, _Traits >::setstate().

getline() [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
             basic_istream< _CharT, _Traits > & __is,
             basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Read a line from stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from is into *__str* until '
' is found, the end of the stream is encountered, or str.max_size() is reached. Any previous contents of *__str* are erased. If end of line is encountered, it is extracted but not stored into *__str*.
References getline(), and std::basic_ios< _CharT, _Traits >::widen().

getline() [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::getline (
            basic_istream< _CharT, _Traits > & __is,
            basic_string< _CharT, _Traits, _Alloc > & __str,
            _CharT __delim)
```

Read a line from stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.
__delim	Character marking end of line.

Returns

Reference to the input stream.

Stores characters from *__is* into *__str* until *__delim* is found, the end of the stream is encountered, or str.max_size() is reached. Any previous contents of *__str* are erased. If *__delim* is encountered, it is extracted but not stored into *__str*.
References std::basic_string< _CharT, _Traits, _Alloc >::erase(), std::basic_string< _CharT, _Traits, _Alloc >::max_size(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().
Referenced by getline(), getline(), getline(), and getline().

hex()

```
ios_base & std::hex (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::hex, ios_base::basefield).
References std::ios_base::basefield, and std::ios_base::hex.

hexfloat()

```
ios_base & std::hexfloat (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::fixed|ios_basescientific, ios_base::floatfield).
References std::ios_base::fixed, std::ios_base::floatfield, and std::ios_base::scientific.

inout_ptr()

```
template<typename _Pointer = void, typename _Smart, typename...  _Args>
auto std::inout_ptr (
            _Smart & __s,
            _Args &&...  __args)  [inline]
```

Adapt a smart pointer for functions taking an inout pointer parameter.

Template Parameters

_Pointer	The type of pointer to convert to.

Parameters

__s	The pointer that should take ownership of the result.
__args	Pack of arguments to use when resetting the smart pointer.

Returns

A `std::inout_ptr_t` referring to `__s`.

Since

C++23

References forward().

internal()

```
ios_base & std::internal (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::internal, ios_base::adjustfield).
References std::ios_base::adjustfield, and std::ios_base::internal.

isalnum()

```
template<typename _CharT>
bool std::isalnum (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::alnum, __c).
References use_facet().

isalpha()

```
template<typename _CharT>
bool std::isalpha (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::alpha, __c).
References use_facet().

isblank()

```
template<typename _CharT>
bool std::isblank (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::blank, __c).
References use_facet().

iscntrl()

```
template<typename _CharT>
bool std::iscntrl (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::cntrl, __c).
References use_facet().

isdigit()

```
template<typename _CharT>
bool std::isdigit (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::digit, __c).
References use_facet().

isgraph()

```
template<typename _CharT>
bool std::isgraph (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::graph, __c).
References use_facet().

islower()

```
template<typename _CharT>
bool std::islower (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::lower, __c).
References use_facet().

isprint()

```
template<typename _CharT>
bool std::isprint (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::print, __c).
References use_facet().

ispunct()

```
template<typename _CharT>
bool std::ispunct (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::punct, __c).
References use_facet().

isspace()

```
template<typename _CharT>
bool std::isspace (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::space, __c).
References use_facet().

isupper()

```
template<typename _CharT>
bool std::isupper (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::upper, __c).
References use_facet().

isxdigit()

```
template<typename _CharT>
bool std::isxdigit (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.is(ctype_base::xdigit, __c).
References use_facet().

left()

```
ios_base & std::left (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::left, ios_base::adjustfield).
References std::ios_base::adjustfield, and std::ios_base::left.

noboolalpha()

```
ios_base & std::noboolalpha (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::boolalpha).
References std::ios_base::boolalpha.

noshowbase()

```
ios_base & std::noshowbase (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::showbase).
References std::ios_base::showbase.

noshowpoint()

```
ios_base & std::noshowpoint (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::showpoint).
References std::ios_base::showpoint.

noshowpos()

```
ios_base & std::noshowpos (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::showpos).
References std::ios_base::showpos.

noskipws()

```
ios_base & std::noskipws (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::skipws).
References std::ios_base::skipws.

nounitbuf()

```
ios_base & std::nounitbuf (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::unitbuf).
References std::ios_base::unitbuf.

nouppercase()

```
ios_base & std::nouppercase (
            ios_base & __base)  [inline]
```

Calls base.unsetf(ios_base::uppercase).
References std::ios_base::uppercase.

oct()

```
ios_base & std::oct (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::oct, ios_base::basefield).
References std::ios_base::basefield, and std::ios_base::oct.

operator"!=() [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator!= (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            const multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Based on operator==.

operator"!=() [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator!= (
            const multiset< _Key, _Compare, _Alloc > & __x,
            const multiset< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns !(x == y).

operator"!=() [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator!= (
            const queue< _Tp, _Seq > & __x,
            const queue< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator==.

operator"!=() [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator!= (
            const set< _Key, _Compare, _Alloc > & __x,
            const set< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns !(x == y).

operator"!=() [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator!= (
            const stack< _Tp, _Seq > & __x,
            const stack< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator==.

operator&()

```
template<size_t _Nb>
bitset< _Nb > std::operator& (
            const bitset< _Nb > & __x,
            const bitset< _Nb > & __y)  [inline], [constexpr], [noexcept]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

Returns

A new bitset.

These should be self-explanatory.

operator+() [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
            _CharT __lhs,
            const basic_string< _CharT, _Traits, _Alloc > & __rhs)  [inline], [nodiscard], [constexpr]
```

Concatenate character and string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::get_allocator(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

operator+() [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
            const _CharT * __lhs,
            const basic_string< _CharT, _Traits, _Alloc > & __rhs)  [inline], [nodiscard], [constexpr]
```

Concatenate C string and string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with value of *__lhs* followed by *__rhs*.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::get_allocator(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

operator+() [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            _CharT __rhs)  [inline], [nodiscard], [constexpr]
```

Concatenate string and character.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::get_allocator(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

operator+() [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            const _CharT * __rhs)  [inline], [nodiscard], [constexpr]
```

Concatenate string and C string.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with *__lhs* followed by *__rhs*.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::get_allocator(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

operator+() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > std::operator+ (
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            const basic_string< _CharT, _Traits, _Alloc > & __rhs)  [inline], [nodiscard], [constexpr]
```

Concatenate two strings.

Parameters

__lhs	First string.
__rhs	Last string.

Returns

New string with value of *__lhs* followed by *__rhs*.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::get_allocator(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

operator<() [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator< (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            const multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Multimap ordering relation.

Parameters

_↩ _x	A multimap.

_↩ _y	A multimap of the same type as __x.

Returns

True iff *x* is lexicographically less than *y*.

This is a total ordering relation. It is linear in the size of the multimaps. The elements must be comparable with <. See std::lexicographical_compare() for how the determination is made.

operator<() [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator< (
            const multiset< _Key, _Compare, _Alloc > & __x,
            const multiset< _Key, _Compare, _Alloc > & __y)  [inline]
```

Multiset ordering relation.

Parameters

_↩ _x	A multiset.
_↩ _y	A multiset of the same type as __x.

Returns

True iff __x is lexicographically less than __y.

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with <. See std::lexicographical_compare() for how the determination is made.

operator<() [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator< (
            const queue< _Tp, _Seq > & __x,
            const queue< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Queue ordering relation.

Parameters

_↩ _x	A queue.
_↩ _y	A queue of the same type as *x*.

Returns

True iff __x is lexicographically less than __y.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with <, and std::↩ lexicographical_compare() is usually used to make the determination.
References std::queue< _Tp, _Sequence >::c.

operator<() [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator< (
            const set< _Key, _Compare, _Alloc > & __x,
            const set< _Key, _Compare, _Alloc > & __y)  [inline]
```

Set ordering relation.

Parameters

_↩ _x	A set.
_↩ _y	A set of the same type as *x*.

Returns

True iff __x is lexicographically less than __y.

This is a total ordering relation. It is linear in the size of the sets. The elements must be comparable with <.
See std::lexicographical_compare() for how the determination is made.

operator<() [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator< (
            const stack< _Tp, _Seq > & __x,
            const stack< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Stack ordering relation.

Parameters

_↩ _x	A stack.
_↩ _y	A stack of the same type as *x*.

Returns

True iff *x* is lexicographically less than __y.

This is an total ordering relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, the elements must be comparable with <, and std::↩ lexicographical_compare() is usually used to make the determination.

operator<<() [1/14]

```
template<typename _Ostream, typename _Tp>
__rvalue_stream_insertion_t< _Ostream, _Tp > std::operator<< (
            _Ostream && __os,
            const _Tp & __x)  [inline]
```

Generic inserter for rvalue stream.

Parameters

__os	An input stream.
__x	A reference to the object being inserted.

Returns

__os

This is just a forwarding function to allow insertion to rvalue streams since they won't bind to the inserter functions that take an lvalue reference.

operator<<() [2/14]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_ostream< _CharT, _Traits > & std::operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Write string to a stream.

Parameters

__os	Output stream.
__str	String to write out.

Returns

Reference to the output stream.

Output characters of *__str* into os following the same rules as for writing a C string.

operator<<() [3/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_ostream< _CharT, _Traits > & std::operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Write string to a stream.

Parameters

__os	Output stream.

__str	String to write out.

Returns

Reference to the output stream.

Output characters of *__str* into os following the same rules as for writing a C string.

operator<<() [4/14]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
            basic_ostream< _CharT, _Traits > & __out,
            _CharT __c)  [inline]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.
If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

operator<<() [5/14]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
            basic_ostream< _CharT, _Traits > & __out,
            char __c)  [inline]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.
If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

operator<<() [6/14]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
            basic_ostream< _CharT, _Traits > & __out,
            const _CharT * __s)  [inline]
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

__s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at __s, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

operator<<() [7/14]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::operator<< (
            basic_ostream< _CharT, _Traits > & __out,
            const char * __s)
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

__s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at __s, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

operator<<() [8/14]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & __out,
            char __c)  [inline]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). `_↩_out.width(0)` is then called.
If `__c` is of type `char` and the character type of the stream is not `char`, the character is widened before insertion.

operator<<() [9/14]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & __out,
            const char * __s)  [inline]
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

`__s` must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at `__s`, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

operator<<() [10/14]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
```

```
            basic_ostream< char, _Traits > & __out,
            const signed char * __s)  [inline]
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

__s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at __s, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

operator<<() [11/14]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & __out,
            const unsigned char * __s)  [inline]
```

String inserters.

Parameters

__out	An output stream.
__s	A character string.

Returns

out

Precondition

__s must be a non-NULL pointer

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts `traits::length(__s)` characters starting at __s, widened if necessary, followed by any required padding (as determined by [22.2.2.2.2]). `__out.width(0)` is then called.

operator<<() [12/14]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & __out,
            signed char __c)  [inline]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). _↩
_out.width(0) is then called.
If __c is of type char and the character type of the stream is not char, the character is widened before insertion.

operator<<() [13/14]

```
template<typename _Traits>
basic_ostream< char, _Traits > & std::operator<< (
            basic_ostream< char, _Traits > & __out,
            unsigned char __c)  [inline]
```

Character inserters.

Parameters

__out	An output stream.
__c	A character.

Returns

out

Behaves like one of the formatted arithmetic inserters described in std::basic_ostream. After constructing a sentry object with good status, this function inserts a single character and any required padding (as determined by [22.2.2.2.2]). _↩
_out.width(0) is then called.
If __c is of type char and the character type of the stream is not char, the character is widened before insertion.

operator<<() [14/14]

```
template<class _CharT, class _Traits, size_t _Nb>
std::basic_ostream< _CharT, _Traits > & std::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const bitset< _Nb > & __x)
```

Global I/O operators for bitsets.
Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept *0* and *1* characters, and will only extract as many digits as the bitset will hold.
References std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_ios< _CharT, _Traits
Referenced by std::shared_ptr< _State_base >::operator<<().

operator<=() [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator<= (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            const multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Based on operator<.

operator<=() [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator<= (
            const multiset< _Key, _Compare, _Alloc > & __x,
            const multiset< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns !(y < x).

operator<=() [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator<= (
            const queue< _Tp, _Seq > & __x,
            const queue< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator<.

operator<=() [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator<= (
            const set< _Key, _Compare, _Alloc > & __x,
            const set< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns !(y < x).

operator<=() [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator<= (
            const stack< _Tp, _Seq > & __x,
            const stack< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator<.

operator<=>() [1/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
auto std::operator<=> (
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            const _CharT * __rhs) -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
[nodiscard], [constexpr], [noexcept]
```

Three-way comparison of a string and a C string.

Parameters

__lhs	A string.
__rhs	A null-terminated string.

Returns

A value indicating whether __lhs is less than, equal to, greater than, or incomparable with __rhs.

operator<=>() [2/7]

```
template<typename _CharT, typename _Traits, typename _Alloc>
auto std::operator<=> (
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            const basic_string< _CharT, _Traits, _Alloc > & __rhs) -> decltype(__detail::__↩
char_traits_cmp_cat< _Traits >(0))  [nodiscard], [constexpr], [noexcept]
```

Three-way comparison of a string and a C string.

Parameters

__lhs	A string.
__rhs	A null-terminated string.

Returns

A value indicating whether __lhs is less than, equal to, greater than, or incomparable with __rhs.

operator<=>() [3/7]

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
            const deque< _Tp, _Alloc > & __x,
            const deque< _Tp, _Alloc > & __y)  [inline], [nodiscard]
```

Deque ordering relation.

Parameters

_↩ _x	A deque.
_↩ _y	A deque of the same type as __x.

Returns

A value indicating whether __x is less than, equal to, greater than, or incomparable with __y.

See std::lexicographical_compare_three_way() for how the determination is made. This operator is used to synthesize relational operators like < and >= etc.
References std::deque< _Tp, _Alloc >::begin(), std::deque< _Tp, _Alloc >::end(), and std::deque< _Tp, _Alloc >::size().

operator<=>() [4/7]

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
            const forward_list< _Tp, _Alloc > & __x,
            const forward_list< _Tp, _Alloc > & __y)  [inline], [nodiscard]
```

Forward list ordering relation.

Parameters

_↩ _x	A forward_list.
_↩ _y	A forward_list of the same type as __x.

Returns

A value indicating whether __x is less than, equal to, greater than, or incomparable with __y.

See std::lexicographical_compare_three_way() for how the determination is made. This operator is used to synthesize relational operators like < and >= etc.

operator<=>() [5/7]

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
            const list< _Tp, _Alloc > & __x,
            const list< _Tp, _Alloc > & __y)  [inline], [nodiscard]
```

List ordering relation.

Parameters

_↩ _x	A list.
_↩ _y	A list of the same type as __x.

Returns

A value indicating whether __x is less than, equal to, greater than, or incomparable with __y.

See std::lexicographical_compare_three_way() for how the determination is made. This operator is used to synthesize relational operators like < and >= etc.
References std::list< _Tp, _Alloc >::begin(), std::list< _Tp, _Alloc >::end(), and std::list< _Tp, _Alloc >::size().

operator<=>() [6/7]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
__detail::__synth3way_t< pair< const _Key, _Tp > > std::operator<=> (
            const map< _Key, _Tp, _Compare, _Alloc > & __x,
            const map< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Map ordering relation.

Parameters

_↩ _x	A map.
_↩ _y	A map of the same type as x.

Returns

A value indicating whether __x is less than, equal to, greater than, or incomparable with __y.

This is a total ordering relation. It is linear in the size of the maps. The elements must be comparable with <.
See std::lexicographical_compare_three_way() for how the determination is made. This operator is used to synthesize relational operators like < and >= etc.

operator<=>() [7/7]

```
template<typename _Tp, typename _Alloc>
__detail::__synth3way_t< _Tp > std::operator<=> (
            const vector< _Tp, _Alloc > & __x,
            const vector< _Tp, _Alloc > & __y)  [nodiscard], [constexpr]
```

Vector ordering relation.

Parameters

_↩ _x	A vector.
_↩ _y	A vector of the same type as __x.

Returns

A value indicating whether __x is less than, equal to, greater than, or incomparable with __y.

See std::lexicographical_compare_three_way() for how the determination is made. This operator is used to synthesize relational operators like < and >= etc.
References std::vector< _Tp, _Alloc >::begin(), std::vector< _Tp, _Alloc >::end(), and lexicographical_compare_three_way().

operator==() [1/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::operator== (
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            const _CharT * __rhs)  [inline], [nodiscard], [constexpr]
```

Test equivalence of string and C string.

Parameters

__lhs	String.
__rhs	C string.

Returns

True if *__lhs.compare*(*__rhs*) == 0. False otherwise.

References std::basic_string< _CharT, _Traits, _Alloc >::data(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

operator==() [2/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::operator== (
```

```
            const basic_string< _CharT, _Traits, _Alloc > & __lhs,
            const basic_string< _CharT, _Traits, _Alloc > & __rhs)  [inline], [nodiscard], [constexpr],
[noexcept]
```

Test equivalence of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Returns

True if *__lhs.compare*(*__rhs*) == 0. False otherwise.

operator==() [3/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
            const deque< _Tp, _Alloc > & __x,
            const deque< _Tp, _Alloc > & __y)  [inline], [nodiscard]
```

Deque equality comparison.

Parameters

_↩ _x	A deque.
_↩ _y	A deque of the same type as *__x*.

Returns

True iff the size and elements of the deques are equal.

This is an equivalence relation. It is linear in the size of the deques. Deques are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

operator==() [4/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
            const forward_list< _Tp, _Alloc > & __lx,
            const forward_list< _Tp, _Alloc > & __ly)
```

Forward list equality comparison.

Parameters

_↩ _lx	A forward_list
_↩ _ly	A forward_list of the same type as *__lx*.

Returns

True iff the elements of the forward lists are equal.

This is an equivalence relation. It is linear in the number of elements of the forward lists. Deques are considered equivalent if corresponding elements compare equal.
References std::forward_list< _Tp, _Alloc >::cbegin(), and std::forward_list< _Tp, _Alloc >::cend().

operator==() [5/14]

```
template<typename _StateT>
bool std::operator== (
            const fpos< _StateT > & __lhs,
            const fpos< _StateT > & __rhs)  [inline]
```

Test if equivalent to another position.

operator==() [6/14]

```
template<typename _Res, typename...  _Args>
bool std::operator== (
            const function< _Res(_Args...)> & __f,
            nullptr_t )  [inline], [noexcept]
```

Test whether a polymorphic function object wrapper is empty.

Returns

`true` if the wrapper has no target, `false` otherwise

This function will not throw exceptions.

operator==() [7/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
            const list< _Tp, _Alloc > & __x,
            const list< _Tp, _Alloc > & __y)  [inline], [nodiscard]
```

List equality comparison.

Parameters

_↩ _x	A list.
_↩ _y	A list of the same type as __x.

Returns

True iff the size and elements of the lists are equal.

This is an equivalence relation. It is linear in the size of the lists. Lists are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

operator==() [8/14]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator== (
```

```
            const map< _Key, _Tp, _Compare, _Alloc > & __x,
            const map< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Map equality comparison.

Parameters

_↩ _x	A map.
_↩ _y	A map of the same type as *x*.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the maps. Maps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

operator==() [9/14]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator== (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            const multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Multimap equality comparison.

Parameters

_↩ _x	A multimap.
_↩ _y	A multimap of the same type as __x.

Returns

True iff the size and elements of the maps are equal.

This is an equivalence relation. It is linear in the size of the multimaps. Multimaps are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

operator==() [10/14]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator== (
            const multiset< _Key, _Compare, _Alloc > & __x,
            const multiset< _Key, _Compare, _Alloc > & __y)  [inline]
```

Multiset equality comparison.

Parameters

_↩ _x	A multiset.

_↩ _y	A multiset of the same type as __x.

Returns

True iff the size and elements of the multisets are equal.

This is an equivalence relation. It is linear in the size of the multisets. Multisets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

operator==() [11/14]

```
template<typename _Tp, typename _Seq>
bool std::operator== (
            const queue< _Tp, _Seq > & __x,
            const queue< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Queue equality comparison.

Parameters

_↩ _x	A queue.
_↩ _y	A queue of the same type as __x.

Returns

True iff the size and elements of the queues are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and queues are considered equivalent if their sequences compare equal.
References std::queue< _Tp, _Sequence >::c.

operator==() [12/14]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator== (
            const set< _Key, _Compare, _Alloc > & __x,
            const set< _Key, _Compare, _Alloc > & __y)  [inline]
```

Set equality comparison.

Parameters

_↩ _x	A set.
_↩ _y	A set of the same type as *x*.

Returns

True iff the size and elements of the sets are equal.

This is an equivalence relation. It is linear in the size of the sets. Sets are considered equivalent if their sizes are equal, and if corresponding elements compare equal.

operator==() [13/14]

```
template<typename _Tp, typename _Seq>
bool std::operator== (
            const stack< _Tp, _Seq > & __x,
            const stack< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Stack equality comparison.

Parameters

_↩ _x	A stack.
_↩ _y	A stack of the same type as __x.

Returns

True iff the size and elements of the stacks are equal.

This is an equivalence relation. Complexity and semantics depend on the underlying sequence type, but the expected rules are: this relation is linear in the size of the sequences, and stacks are considered equivalent if their sequences compare equal.

operator==() [14/14]

```
template<typename _Tp, typename _Alloc>
bool std::operator== (
            const vector< _Tp, _Alloc > & __x,
            const vector< _Tp, _Alloc > & __y)  [inline], [nodiscard], [constexpr]
```

Vector equality comparison.

Parameters

_↩ _x	A vector.
_↩ _y	A vector of the same type as __x.

Returns

True iff the size and elements of the vectors are equal.

This is an equivalence relation. It is linear in the size of the vectors. Vectors are considered equivalent if their sizes are equal, and if corresponding elements compare equal.
References std::vector< _Tp, _Alloc >::begin(), std::vector< _Tp, _Alloc >::end(), and std::vector< _Tp, _Alloc >::size().

operator>() [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator> (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            const multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Based on operator<.

operator>() [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator> (
            const multiset< _Key, _Compare, _Alloc > & __x,
            const multiset< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns y < x.

operator>() [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator> (
            const queue< _Tp, _Seq > & __x,
            const queue< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator<.

operator>() [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator> (
            const set< _Key, _Compare, _Alloc > & __x,
            const set< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns y < x.

operator>() [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator> (
            const stack< _Tp, _Seq > & __x,
            const stack< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator<.

operator>=() [1/5]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
bool std::operator>= (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            const multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline]
```

Based on operator<.

operator>=() [2/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator>= (
            const multiset< _Key, _Compare, _Alloc > & __x,
            const multiset< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns !(x < y).

operator>=() [3/5]

```
template<typename _Tp, typename _Seq>
bool std::operator>= (
            const queue< _Tp, _Seq > & __x,
            const queue< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator<.

operator>=() [4/5]

```
template<typename _Key, typename _Compare, typename _Alloc>
bool std::operator>= (
            const set< _Key, _Compare, _Alloc > & __x,
            const set< _Key, _Compare, _Alloc > & __y)  [inline]
```

Returns !(x < y).

operator>=() [5/5]

```
template<typename _Tp, typename _Seq>
bool std::operator>= (
            const stack< _Tp, _Seq > & __x,
            const stack< _Tp, _Seq > & __y)  [inline], [nodiscard]
```

Based on operator<.

operator>>() [1/10]

```
template<typename _Istream, typename _Tp>
__rvalue_stream_extraction_t< _Istream, _Tp > std::operator>> (
            _Istream && __is,
            _Tp && __x)  [inline]
```

Generic extractor for rvalue stream.

Parameters

_↩ _is	An input stream.
_↩ _x	A reference to the extraction target.

Returns

__is

This is just a forwarding function to allow extraction from rvalue streams since they won't bind to the extractor functions that take an lvalue reference.
References forward(), and move().

operator>>() [2/10]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::operator>> (
            basic_istream< _CharT, _Traits > & __in,
            _CharT & __c)
```

Character extractors.

Parameters

_↩ _*in*	An input stream.
_↩ _*c*	A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in std::basic_istream. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in __*c*. Otherwise, sets failbit in the input stream.
References std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rd and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [3/10]

```
template<typename _CharT, typename _Traits, size_t _Num>
basic_istream< _CharT, _Traits > & std::operator>> (
            basic_istream< _CharT, _Traits > & __in,
            _CharT(&) __s[_Num])  [inline]
```

Character string extractors.

Parameters

_↩ _*in*	An input stream.
_↩ _*s*	A character array (or a pointer to an array before C++20).

Returns

__in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array ___`s`. `n` is defined as:

- if `width()` is greater than zero, `n` is `min(width(), n)`
- otherwise `n` is the number of elements of the array
- (before C++20 the pointer is assumed to point to an array of the largest possible size for an array of `char_type`).

Characters are extracted and stored until one of the following happens:

- `n - 1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale

`width(0)` is then called for the input stream.
If no characters are extracted, sets failbit.

operator>>() [4/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
basic_istream< _CharT, _Traits > & std::operator>> (
            basic_istream< _CharT, _Traits > & __is,
            __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > & __str)
```

Read stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from *__is* into *__str* until whitespace is found, the end of the stream is encountered, or str.max_size() is reached. If is.width() is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased.
References std::ios_base::getloc(), std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), use_facet(), and std::ios_base::width().

operator>>() [5/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_istream< _CharT, _Traits > & std::operator>> (
            basic_istream< _CharT, _Traits > & __is,
            basic_string< _CharT, _Traits, _Alloc > & __str)
```

Read stream into a string.

Parameters

__is	Input stream.
__str	Buffer to store into.

Returns

Reference to the input stream.

Stores characters from *__is* into *__str* until whitespace is found, the end of the stream is encountered, or str.max_size() is reached. If is.width() is non-zero, that is the limit on the number of characters stored into *__str*. Any previous contents of *__str* are erased.
References std::basic_string< _CharT, _Traits, _Alloc >::append(), std::basic_string< _CharT, _Traits, _Alloc >::erase(), std::ios_base::getloc(), std::basic_string< _CharT, _Traits, _Alloc >::max_size(), std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), use_facet(), and std::ios_base::width().

operator>>() [6/10]

```
template<class _Traits>
basic_istream< char, _Traits > & std::operator>> (
```

```
            basic_istream< char, _Traits > & __in,
            signed char & __c)  [inline]
```

Character extractors.

Parameters

_↩ _in	An input stream.
_↩ _c	A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in std::basic_istream. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in __c. Otherwise, sets failbit in the input stream.

operator>>() [7/10]

```
template<class _Traits, size_t _Num>
basic_istream< char, _Traits > & std::operator>> (
            basic_istream< char, _Traits > & __in,
            signed char(&) __s[_Num])  [inline]
```

Character string extractors.

Parameters

_↩ _in	An input stream.
_↩ _s	A character array (or a pointer to an array before C++20).

Returns

__in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array __s. `n` is defined as:

- if `width()` is greater than zero, `n` is `min(width(), n)`
- otherwise `n` is the number of elements of the array
- (before C++20 the pointer is assumed to point to an array of the largest possible size for an array of `char_type`).

Characters are extracted and stored until one of the following happens:

- `n - 1` characters are stored
- EOF is reached
- the next character is whitespace according to the current locale

`width(0)` is then called for the input stream.
If no characters are extracted, sets failbit.

operator>>() [8/10]

```
template<class _Traits>
basic_istream< char, _Traits > & std::operator>> (
            basic_istream< char, _Traits > & __in,
            unsigned char & __c)  [inline]
```

Character extractors.

Parameters

_↩ _in	An input stream.
_↩ _c	A character reference.

Returns

in

Behaves like one of the formatted arithmetic extractors described in std::basic_istream. After constructing a sentry object with good status, this function extracts a character (if one is available) and stores it in __c. Otherwise, sets failbit in the input stream.

operator>>() [9/10]

```
template<class _Traits, size_t _Num>
basic_istream< char, _Traits > & std::operator>> (
            basic_istream< char, _Traits > & __in,
            unsigned char(&) __s[_Num])  [inline]
```

Character string extractors.

Parameters

_↩ _in	An input stream.
_↩ _s	A character array (or a pointer to an array before C++20).

Returns

__in

Behaves like one of the formatted arithmetic extractors described in `std::basic_istream`. After constructing a sentry object with good status, this function extracts up to `n` characters and stores them into the array ___s. `n` is defined as:

- if `width()` is greater than zero, `n` is `min(width(), n)`
- otherwise `n` is the number of elements of the array
- (before C++20 the pointer is assumed to point to an array of the largest possible size for an array of `char_type`).

Characters are extracted and stored until one of the following happens:

- `n - 1` characters are stored

- EOF is reached
- the next character is whitespace according to the current locale

`width(0)` is then called for the input stream.
If no characters are extracted, sets failbit.

operator>>() [10/10]

```
template<class _CharT, class _Traits, size_t _Nb>
std::basic_istream< _CharT, _Traits > & std::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            bitset< _Nb > & __x)
```

Global I/O operators for bitsets.
Direct I/O between streams and bitsets is supported. Output is straightforward. Input will skip whitespace, only accept *0* and *1* characters, and will only extract as many digits as the bitset will hold.

operator^()

```
template<size_t _Nb>
bitset< _Nb > std::operator^ (
            const bitset< _Nb > & __x,
            const bitset< _Nb > & __y)  [inline], [constexpr], [noexcept]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

Returns

A new bitset.

These should be self-explanatory.

operator"" | ()

```
template<size_t _Nb>
bitset< _Nb > std::operator| (
            const bitset< _Nb > & __x,
            const bitset< _Nb > & __y)  [inline], [constexpr], [noexcept]
```

Global bitwise operations on bitsets.

Parameters

_↩ _x	A bitset.
_↩ _y	A bitset of the same size as __x.

Returns

A new bitset.

These should be self-explanatory.

out_ptr()

```
template<typename _Pointer = void, typename _Smart, typename...  _Args>
auto std::out_ptr (
            _Smart & __s,
            _Args &&...  __args)  [inline]
```

Adapt a smart pointer for functions taking an output pointer parameter.

Template Parameters

_Pointer	The type of pointer to convert to.

Parameters

__s	The pointer that should take ownership of the result.
__args	Pack of arguments to use when resetting the smart pointer.

Returns

A `std::out_ptr_t` referring to `__s`.

Since

C++23

References forward().

put_money()

```
template<typename _MoneyT>
_Put_money< _MoneyT > std::put_money (
            const _MoneyT & __mon,
            bool __intl = false)  [inline]
```

Extended manipulator for inserting money.

Parameters

__mon	Either long double or a specialization of `basic_string`.
__intl	A bool indicating whether international format is to be used.

Sent to a stream object, this manipulator inserts *__mon*.

put_time()

```
template<typename _CharT>
_Put_time< _CharT > std::put_time (
```

```
            const std::tm * __tmb,
            const _CharT * __fmt)  [inline]
```

Extended manipulator for formatting time.
This manipulator uses time_put::put to format time. [ext.manip]

Parameters

__tmb	struct tm time data to format.
__fmt	format string.

rbegin() [1/4]

```
template<typename _Container>
auto std::rbegin (
            _Container & __cont) -> decltype(__cont.rbegin())  [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the container.

Parameters

__cont	Container.

Since

C++14

Referenced by crbegin().

rbegin() [2/4]

```
template<typename _Tp, size_t _Nm>
reverse_iterator< _Tp * > std::rbegin (
            _Tp(&) __arr[_Nm])  [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the array.

Parameters

__arr	Array.

Since

C++14

rbegin() [3/4]

```
template<typename _Container>
auto std::rbegin (
            const _Container & __cont) -> decltype(__cont.rbegin())  [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the const container.

Parameters

__cont	Container.

Since

C++14

rbegin() [4/4]

```
template<typename _Tp>
reverse_iterator< const _Tp * > std::rbegin (
            initializer_list< _Tp > __il)  [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing to the last element of the initializer_list.

Parameters

_↩ _il	initializer_list.

Since

C++14

rend() [1/4]

```
template<typename _Container>
auto std::rend (
            _Container & __cont) -> decltype(__cont.rend())  [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the container.

Parameters

__cont	Container.

Since

C++14

Referenced by crend().

rend() [2/4]

```
template<typename _Tp, size_t _Nm>
reverse_iterator< _Tp * > std::rend (
            _Tp(&) __arr[_Nm])  [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the array.

Parameters

__arr	Array.

Since

C++14

rend() [3/4]

```
template<typename _Container>
auto std::rend (
            const _Container & __cont) -> decltype(__cont.rend())  [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the const container.

Parameters

__cont	Container.

Since

C++14

rend() [4/4]

```
template<typename _Tp>
reverse_iterator< const _Tp * > std::rend (
            initializer_list< _Tp > __il)  [inline], [constexpr], [noexcept]
```

Return a reverse iterator pointing one past the first element of the initializer_list.

Parameters

_↩ _il	initializer_list.

Since

C++14

replace_copy()

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

Copy a sequence, replacing each element of one value with another value.

Parameters

__first	An input iterator.
__last	An input iterator.

__result	An output iterator.
__old_value	The value to be replaced.
__new_value	The replacement value.

Returns

The end of the output sequence, `result+`(last-first).

Copies each element in the input range [__first,__last) to the output range [__result,__result+(__last-__first)) replacing elements equal to `__old_value` with `__new_value`.

resetiosflags()

```
_Resetiosflags std::resetiosflags (
            ios_base::fmtflags __mask)  [inline]
```

Manipulator for `setf`.

Parameters

__mask	A format flags mask.

Sent to a stream object, this manipulator resets the specified flags, via *stream.setf(0,__mask)*.

return_temporary_buffer()

```
template<typename _Tp>
void std::return_temporary_buffer (
            _Tp * __p)  [inline]
```

The companion to get_temporary_buffer().

Parameters

__p	A buffer previously allocated by get_temporary_buffer.

Frees the memory pointed to by __p.

right()

```
ios_base & std::right (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::right, ios_base::adjustfield).
References std::ios_base::adjustfield, and std::ios_base::right.

sample()

```
template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _↩
UniformRandomBitGenerator>
_SampleIterator std::sample (
            _PopulationIterator __first,
            _PopulationIterator __last,
            _SampleIterator __out,
```

```
            _Distance __n,
            _UniformRandomBitGenerator && __g)
```

Take a random sample from a population.
References forward().

saturating_add()

```
template<typename _Tp>
_Tp std::saturating_add (
            _Tp __x,
            _Tp __y)  [constexpr], [noexcept]
```

Add two integers, with saturation in case of overflow.

saturating_cast()

```
template<typename _Res, typename _Tp>
_Res std::saturating_cast (
            _Tp __x)  [constexpr], [noexcept]
```

Divide one integer by another, with saturation in case of overflow.

saturating_div()

```
template<typename _Tp>
_Tp std::saturating_div (
            _Tp __x,
            _Tp __y)  [constexpr], [noexcept]
```

Divide one integer by another, with saturation in case of overflow.

saturating_mul()

```
template<typename _Tp>
_Tp std::saturating_mul (
            _Tp __x,
            _Tp __y)  [constexpr], [noexcept]
```

Multiply two integers, with saturation in case of overflow.

saturating_sub()

```
template<typename _Tp>
_Tp std::saturating_sub (
            _Tp __x,
            _Tp __y)  [constexpr], [noexcept]
```

Subtract one integer from another, with saturation in case of overflow.

scientific()

```
ios_base & std::scientific (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::scientific, ios_base::floatfield).
References std::ios_base::floatfield, and std::ios_base::scientific.

search()

```
template<typename _ForwardIterator, typename _Searcher>
_ForwardIterator std::search (
```

```
            _ForwardIterator __first,
            _ForwardIterator __last,
            const _Searcher & __searcher)  [inline], [nodiscard], [constexpr]
```

Search a sequence using a Searcher object.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__searcher	A callable object.

Returns

`__searcher(__first,__last).first`

set_new_handler()

```
new_handler std::set_new_handler (
            new_handler ) throw ( )
```

Takes a replacement handler as the argument, returns the previous handler.

setbase()

```
_Setbase std::setbase (
            int __base)  [inline]
```

Manipulator for `setf`.

Parameters

__base	A numeric base.

Sent to a stream object, this manipulator changes the `ios_base::basefield` flags to `oct`, `dec`, or `hex` when *base* is 8, 10, or 16, accordingly, and to 0 if *__base* is any other value.

setfill()

```
template<typename _CharT>
_Setfill< _CharT > std::setfill (
            _CharT __c)  [inline]
```

Manipulator for `fill`.

Parameters

_↩ _c	The new fill character.

Sent to a stream object, this manipulator calls `fill(__c)` for that object.

setiosflags()

```
_Setiosflags std::setiosflags (
            ios_base::fmtflags __mask)  [inline]
```

Manipulator for `setf`.

Parameters

__mask	A format flags mask.

Sent to a stream object, this manipulator sets the format flags to *__mask*.

setprecision()

```
_Setprecision std::setprecision (
            int __n)  [inline]
```

Manipulator for `precision`.

Parameters

_↩ _n	The new precision.

Sent to a stream object, this manipulator calls `precision(__n)` for that object.

setw()

```
_Setw std::setw (
            int __n)  [inline]
```

Manipulator for `width`.

Parameters

_↩ _n	The new width.

Sent to a stream object, this manipulator calls `width(__n)` for that object.

showbase()

```
ios_base & std::showbase (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::showbase).
References std::ios_base::showbase.

showpoint()

```
ios_base & std::showpoint (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::showpoint).
References std::ios_base::showpoint.

showpos()

```
ios_base & std::showpos (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::showpos).
References std::ios_base::showpos.

size() [1/2]

```
template<typename _Container>
auto std::size (
            const _Container & __cont) -> decltype(__cont.size())  [nodiscard], [constexpr],
[noexcept]
```

Return the size of a container.

Parameters

__cont	Container.

Since

C++17

Referenced by std::basic_stringbuf< char >::basic_stringbuf(), std::deque< _Tp, _Alloc >::_M_new_elements_at_back(), std::deque< _Tp, _Alloc >::_M_new_elements_at_front(), std::deque< _Tp, _Alloc >::_M_push_back_aux(), std::deque< _Tp, _Alloc >, std::basic_string< _CharT, _Traits, _Alloc >::find(), std::basic_string< _CharT, _Traits, _Alloc >::find(), std::basic_string< _CharT, _Traits, std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_first_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of(), std::basic_string< _CharT, _Traits, _Alloc >::find_last_of(), std::deque< _Tp, _Alloc >::operator=(), std::vector< _Tp, _Alloc >::operator, std::basic_string< _CharT, _Traits, _Alloc >::reserve(), std::vector< _Tp, _Alloc >::reserve(), std::vector< _State< _TraitsT::char_type >, std::basic_string< _CharT, _Traits, _Alloc >::rfind(), and std::basic_string< _CharT, _Traits, _Alloc >::rfind().

size() [2/2]

```
template<typename _Tp, size_t _Nm>
size_t std::size (
            const  _Tp(&)[_Nm])  [nodiscard], [constexpr], [noexcept]
```

Return the size of an array.

Parameters

-	An array.

Returns

The number of elements in the array.

Since

C++17

skipws()

```
ios_base & std::skipws (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::skipws).
References std::ios_base::skipws.

ssize() [1/2]

```
template<typename _Container>
auto std::ssize (
            const _Container & __cont) -> common_type_t< ptrdiff_t, make_signed_t< decltype(__↩
cont.size())> > [nodiscard], [constexpr], [noexcept]
```

Return the size of a container, as a signed integer.

Parameters

__cont	Container.

Since

C++20

ssize() [2/2]

```
template<typename _Tp, ptrdiff_t _Num>
ptrdiff_t std::ssize (
            const _Tp(&)[_Num]) [nodiscard], [constexpr], [noexcept]
```

Return the length of an array, as a signed integer.

Since

C++20

static_pointer_cast()

```
template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
__shared_ptr< _Tp, _Lp > std::static_pointer_cast (
            const __shared_ptr< _Tp1, _Lp > & __r) [inline], [noexcept]
```

static_pointer_cast

swap() [1/19]

```
template<class _CharT, class _Traits>
void std::swap (
            basic_filebuf< _CharT, _Traits > & __x,
            basic_filebuf< _CharT, _Traits > & __y) [inline]
```

Swap specialization for filebufs.

swap() [2/19]

```
template<class _CharT, class _Traits>
void std::swap (
            basic_fstream< _CharT, _Traits > & __x,
            basic_fstream< _CharT, _Traits > & __y) [inline]
```

Swap specialization for fstreams.

swap() [3/19]

```
template<class _CharT, class _Traits>
void std::swap (
```

```
            basic_ifstream< _CharT, _Traits > & __x,
            basic_ifstream< _CharT, _Traits > & __y)  [inline]
```

Swap specialization for ifstreams.

swap() [4/19]

```
template<class _CharT, class _Traits, class _Allocator>
void std::swap (
            basic_istringstream< _CharT, _Traits, _Allocator > & __x,
            basic_istringstream< _CharT, _Traits, _Allocator > & __y)  [inline]
```

Swap specialization for istringstreams.

swap() [5/19]

```
template<class _CharT, class _Traits>
void std::swap (
            basic_ofstream< _CharT, _Traits > & __x,
            basic_ofstream< _CharT, _Traits > & __y)  [inline]
```

Swap specialization for ofstreams.

swap() [6/19]

```
template<class _CharT, class _Traits, class _Allocator>
void std::swap (
            basic_ostringstream< _CharT, _Traits, _Allocator > & __x,
            basic_ostringstream< _CharT, _Traits, _Allocator > & __y)  [inline]
```

Swap specialization for ostringstreams.

swap() [7/19]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::swap (
            basic_string< _CharT, _Traits, _Alloc > & __lhs,
            basic_string< _CharT, _Traits, _Alloc > & __rhs)  [inline], [constexpr], [noexcept]
```

Swap contents of two strings.

Parameters

__lhs	First string.
__rhs	Second string.

Exchanges the contents of *__lhs* and *__rhs* in constant time.

swap() [8/19]

```
template<class _CharT, class _Traits, class _Allocator>
void std::swap (
            basic_stringbuf< _CharT, _Traits, _Allocator > & __x,
            basic_stringbuf< _CharT, _Traits, _Allocator > & __y)  [inline], [noexcept]
```

Swap specialization for stringbufs.

swap() [9/19]

```
template<class _CharT, class _Traits, class _Allocator>
void std::swap (
```

```
            basic_stringstream< _CharT, _Traits, _Allocator > & __x,
            basic_stringstream< _CharT, _Traits, _Allocator > & __y)  [inline]
```

Swap specialization for stringstreams.

swap() [10/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
            deque< _Tp, _Alloc > & __x,
            deque< _Tp, _Alloc > & __y)  [inline], [noexcept]
```

See std::deque::swap().

swap() [11/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
            forward_list< _Tp, _Alloc > & __lx,
            forward_list< _Tp, _Alloc > & __ly)  [inline], [noexcept]
```

See std::forward_list::swap().

swap() [12/19]

```
template<typename _Res, typename...  _Args>
void std::swap (
            function< _Res(_Args...)> & __x,
            function< _Res(_Args...)> & __y)  [inline], [noexcept]
```

Swap the targets of two polymorphic function object wrappers.
This function will not throw exceptions.

swap() [13/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
            list< _Tp, _Alloc > & __x,
            list< _Tp, _Alloc > & __y)  [inline], [noexcept]
```

See std::list::swap().

swap() [14/19]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
void std::swap (
            map< _Key, _Tp, _Compare, _Alloc > & __x,
            map< _Key, _Tp, _Compare, _Alloc > & __y)  [inline], [noexcept]
```

See std::map::swap().
References std::pair< _T1, _T2 >::swap().

swap() [15/19]

```
template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
void std::swap (
            multimap< _Key, _Tp, _Compare, _Alloc > & __x,
            multimap< _Key, _Tp, _Compare, _Alloc > & __y)  [inline], [noexcept]
```

See std::multimap::swap().
References std::multimap< _Key, _Tp, _Compare, _Alloc >::swap().

swap() [16/19]

```
template<typename _Key, typename _Compare, typename _Alloc>
void std::swap (
            multiset< _Key, _Compare, _Alloc > & __x,
            multiset< _Key, _Compare, _Alloc > & __y)  [inline], [noexcept]
```

See std::multiset::swap().
References std::multiset< _Key, _Compare, _Alloc >::swap().

swap() [17/19]

```
template<typename _Key, typename _Compare, typename _Alloc>
void std::swap (
            set< _Key, _Compare, _Alloc > & __x,
            set< _Key, _Compare, _Alloc > & __y)  [inline], [noexcept]
```

See std::set::swap().
References std::set< _Key, _Compare, _Alloc >::swap().

swap() [18/19]

```
template<typename _Tp, typename _Alloc>
void std::swap (
            vector< _Tp, _Alloc > & __x,
            vector< _Tp, _Alloc > & __y)  [inline], [constexpr], [noexcept]
```

See std::vector::swap().

tolower()

```
template<typename _CharT>
_CharT std::tolower (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.tolower(__c).
References tolower(), and use_facet().
Referenced by tolower().

toupper()

```
template<typename _CharT>
_CharT std::toupper (
            _CharT __c,
            const locale & __loc)  [inline]
```

Convenience interface to ctype.toupper(__c).
References toupper(), and use_facet().
Referenced by toupper().

unitbuf()

```
ios_base & std::unitbuf (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::unitbuf).
References std::ios_base::unitbuf.

uppercase()

```
ios_base & std::uppercase (
            ios_base & __base)  [inline]
```

Calls base.setf(ios_base::uppercase).
References std::ios_base::uppercase.

ws()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::ws (
            basic_istream< _CharT, _Traits > & __is)
```

Quick and easy way to eat whitespace.
This manipulator extracts whitespace characters, stopping when the next character is non-whitespace, or when the input sequence is empty. If the sequence is empty, `eofbit` is set in the stream, but not `failbit`.
The current locale is used to distinguish whitespace characters.
Example:

```
MyClass  mc;

std::cin » std::ws » mc;
```

will skip leading whitespace before calling operator>> on cin and your object. Note that the same effect can be achieved by creating a std::basic_istream::sentry inside your definition of operator>>.
References std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::getloc(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::
std::basic_ios< _CharT, _Traits >::setstate(), and use_facet().

4.10.5 Variable Documentation

cerr

```
ostream std::cerr  [extern]
```

Linked to standard error (unbuffered).

cin

```
istream std::cin  [extern]
```

Linked to standard input.

clog

```
ostream std::clog  [extern]
```

Linked to standard error (buffered).

cout

```
ostream std::cout  [extern]
```

Linked to standard output.

default_sentinel

```
default_sentinel_t std::default_sentinel  [inline], [constexpr]
```

A default sentinel value.

destroying_delete

```
destroying_delete_t std::destroying_delete  [inline], [constexpr]
```

Tag variable of type destroying_delete_t.

ignore

```
_Swallow_assign std::ignore  [inline], [constexpr]
```

Used with `std::tie` to ignore an element of a tuple

When using `std::tie` to assign the elements of a tuple to variables, unwanted elements can be ignored by using `std::ignore`. For example:

```
int x, y;
std::tie(x, std::ignore, y) = std::make_tuple(1, 2, 3);
```

This assignment will perform `x=1; std::ignore=2; y=3;` which results in the second element being ignored.

Since

C++11

wcerr

```
wostream std::wcerr  [extern]
```

Linked to standard error (unbuffered).

wcin

```
wistream std::wcin  [extern]
```

Linked to standard input.

wclog

```
wostream std::wclog  [extern]
```

Linked to standard error (buffered).

wcout

```
wostream std::wcout  [extern]
```

Linked to standard output.

4.11 std::__debug Namespace Reference

Classes

- class bitset
- class deque
- class forward_list
- class list
- class map
- class multimap
- class multiset
- class set
- class unordered_map
- class unordered_multimap
- class unordered_multiset
- class unordered_set
- class vector

Functions

- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  deque (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> deque< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  deque (size_t, _Tp, _Allocator=_Allocator()) -> deque< _Tp, _Allocator >
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  forward_list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward_list< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  forward_list (size_t, _Tp, _Allocator=_Allocator()) -> forward_list< _Tp, _Allocator >
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> list< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  list (size_t, _Tp, _Allocator=_Allocator()) -> list< _Tp, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  map (_InputIterator, _InputIterator, _Allocator) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _Input↩ Iterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter↩ _to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  map (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> map< __iter_key_t< ↩ _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> map< _Key, _Tp, less< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  map (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> map< _Key, _Tp, _Compare, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  multimap (_InputIterator, _InputIterator, _Allocator) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter↩ _to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multimap (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< __iter↩ _key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> multimap< _Key, _Tp, less< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multimap (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< _Key, _Tp, _Compare, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  multiset (_InputIterator, _InputIterator, _Allocator) -> multiset< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >

- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multiset (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  multiset (initializer_list< _Key >, _Allocator) -> multiset< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  multiset (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< _Key, _Compare, _Allocator >
- template<size_t _Nb>
  constexpr bitset< _Nb > **operator&** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<typename _CharT, typename _Traits, size_t _Nb>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **operator**<=> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **operator**<=> (const forward_list< _Tp, _Alloc > &__x, const forward_list< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **operator**<=> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< pair< const _Key, _Tp > > **operator**<=> (const map< _Key, _Tp, _Compare, _Alloc > &__lhs, const map< _Key, _Tp, _Compare, _Alloc > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< pair< const _Key, _Tp > > **operator**<=> (const multimap< _Key, _Tp, _Compare, _Alloc > &__lhs, const multimap< _Key, _Tp, _Compare, _Alloc > &__rhs)
- template<typename _Key, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< _Key > **operator**<=> (const multiset< _Key, _Compare, _Alloc > &__lhs, const multiset< _Key, _Compare, _Alloc > &__rhs)
- template<typename _Key, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< _Key > **operator**<=> (const set< _Key, _Compare, _Alloc > &__lhs, const set< _Key, _Compare, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **operator**<=> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool **operator==** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc>
  bool **operator==** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp, typename _Alloc>
  bool **operator==** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  bool **operator==** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  bool **operator==** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)

- template<typename _Key, typename _Compare, typename _Allocator>
  bool **operator==** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Compare, typename _Allocator>
  bool **operator==** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  bool **operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  bool **operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  bool **operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  bool **operator==** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  constexpr bool **operator==** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, size_t _Nb>
  std::basic_istream< _CharT, _Traits > & **operator>>** (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)
- template<size_t _Nb>
  constexpr bitset< _Nb > **operator^** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  constexpr bitset< _Nb > **operator|** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  set (initializer_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  set (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _Compare, _Allocator >
- template<typename _Tp, typename _Alloc>
  void **swap** (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)
- template<typename _Tp, typename _Alloc>
  void swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))
- template<typename _Tp, typename _Alloc>
  void **swap** (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  void **swap** (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)

- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  void **swap** (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)
- template<typename _Key, typename _Compare, typename _Allocator>
  void **swap** (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */)
- template<typename _Key, typename _Compare, typename _Allocator>
  void **swap** (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  void **swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  void **swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  void **swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  void **swap** (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp, typename _Alloc>
  constexpr void **swap** (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>

unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >

- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< _Key, _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< _Key, _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>

unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >

- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (initializer_list< _Tp >, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >

- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  vector (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> vector< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  vector (size_t, _Tp, _Allocator=_Allocator()) -> vector< _Tp, _Allocator >

4.11.1 Detailed Description

GNU debug code, replaces standard behavior with debug behavior.
Macros and namespaces used by the implementation outside of debug wrappers to verify certain properties. The _glibcxx_requires_xxx macros are merely wrappers around the __glibcxx_check_xxx wrappers when we are compiling with debug mode, but disappear when we are in release mode so that there is no checking performed in, e.g., the standard library algorithms.

4.11.2 Function Documentation

swap()

```
template<typename _Tp, typename _Alloc>
void std::__debug::swap (
            forward_list< _Tp, _Alloc > & __lx,
            forward_list< _Tp, _Alloc > & __ly)  [inline], [noexcept]
```

See std::forward_list::swap().

4.12 std::__detail Namespace Reference

Classes

- struct _BracketMatcher
- class _Compiler
- class _Executor
- struct _List_node_base
- struct _List_node_header
- struct _Quoted_string
- class _Scanner
- class _StateSeq

Typedefs

- template<typename _Tp, typename _Up>
  using **__cmp3way_res_t**
- template<typename _Tp>
  using **__cref**
- template<typename _Iter>
  using **__iter_concept**
- template<typename _Tp>
  using **__iter_diff_t**
- template<typename _Iter, typename _Tp = _Iter>
  using **__iter_traits**
- template<typename _Tp>
  using **__iter_value_t**
- template<typename _Tp>
  using **__piecewise_distributions_storage_t**
- template<typename _Tp>
  using **__range_iter_t**
- template<typename _Tp>
  using **__unsigned_least_t**
- template<typename _Tp>
  using **__with_ref**
- template<typename _CharT>
  using **_Matcher**
- typedef long **_StateIdT**

Enumerations

- enum **_ExecutorFrameOpcode** : unsigned char {
  _S_fopcode_next , **_S_fopcode_fallback_next** , **_S_fopcode_rep_once_more** , **_S_fopcode_fallback_↩rep_once_more** ,
  _S_fopcode_posix_alternative , **_S_fopcode_merge_sol** , **_S_fopcode_restore_cur_results** , **_S_↩fopcode_restore_rep_count** ,
  _S_fopcode_decrement_rep_count }
- enum _Opcode : int {
  _S_opcode_unknown , **_S_opcode_alternative** , **_S_opcode_repeat** , **_S_opcode_backref** ,
  _S_opcode_line_begin_assertion , **_S_opcode_line_end_assertion** , **_S_opcode_word_boundary** , **_S_↩opcode_subexpr_lookahead** ,
  _S_opcode_subexpr_begin , **_S_opcode_subexpr_end** , **_S_opcode_dummy** , **_S_opcode_match** ,
  _S_opcode_accept }
- enum class **_RegexExecutorPolicy** : int { **_S_auto** , **_S_alternate** }

Functions

- template<typename _Res, typename _Tp>
  constexpr _Res **__abs_r** (_Tp __val)
- template<typename>
  void **__abs_r** (bool)=delete
- template<typename _ChTraits>
  constexpr auto **__char_traits_cmp_cat** (int __cmp) noexcept
- template<typename... _Ts>
  constexpr auto **__common_cmp_cat** ()

- template<typename _Container, typename _UnsafeContainer, typename _Predicate>
  constexpr _Container::size_type **__erase_if** (_Container &__cont, _UnsafeContainer &__ucont, _Predicate __pred)
- template<typename _Container, typename _UnsafeContainer, typename _Predicate>
  _Container::size_type **__erase_nodes_if** (_Container &__cont, _UnsafeContainer &__ucont, _Predicate __pred)
- template<typename _ValT, typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **__extract_params** (basic_istream< _CharT, _Traits > &__is, vector< _ValT > &__vals, size_t __n)
- template<bool _DecOnly, typename _Tp>
  constexpr bool __from_chars_alnum (const char *&__first, const char *__last, _Tp &__val, int __base)
- template<bool _DecOnly = false>
  constexpr unsigned char **__from_chars_alnum_to_val** (unsigned char __c)
- template<bool _DecOnly, typename _Tp>
  constexpr bool __from_chars_pow2_base (const char *&__first, const char *__last, _Tp &__val, int __base)
- template<typename _Tp>
  constexpr _Tp **__gcd** (_Tp __m, _Tp __n)
- template<typename _Tp>
  _Tp * **__get_temporary_buffer** (ptrdiff_t __len) noexcept
- template<typename _InputIterator, typename _RealType>
  bool **__load_first2** (_InputIterator &__first, _InputIterator __last, _RealType *__out)
- template<typename _Tp>
  constexpr bool **__p1_representable_as_double** (_Tp __x) noexcept
- template<typename _Tp>
  constexpr bool **__raise_and_add** (_Tp &__val, int __base, unsigned char __c)
- template<typename _BiIter, typename _Alloc, typename _CharT, typename _TraitsT>
  bool **__regex_algo_impl** (_BiIter __s, _BiIter __e, match_results< _BiIter, _Alloc > &__m, const basic_regex< _CharT, _TraitsT > &__re, regex_constants::match_flag_type __flags, _RegexExecutorPolicy __policy, bool __match_mode)
- template<typename _Tp>
  constexpr bool **__representable_as_double** (_Tp __x) noexcept
- template<typename _Tp>
  void **__return_temporary_buffer** (_Tp *__p, size_t __len)
- template<typename _Tp>
  constexpr to_chars_result **__to_chars** (char *__first, char *__last, _Tp __val, int __base) noexcept
- template<typename _Tp>
  constexpr to_chars_result **__to_chars_10** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr void **__to_chars_10_impl** (char *__first, unsigned __len, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **__to_chars_16** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **__to_chars_2** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **__to_chars_8** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr unsigned **__to_chars_len** (_Tp __value, int __base) noexcept
- template<typename _Tp>
  constexpr unsigned **__to_chars_len_2** (_Tp __value) noexcept
- template<typename _CharT, typename _Traits, typename _String>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Quoted_string< _String, _CharT > &__str)

- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const _Quoted_string< const _CharT *, _CharT > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, const _Quoted_string< basic_string< _CharT, _Traits, _Alloc > &, _CharT > &__str)

Variables

- template<typename _Tp>
  constexpr unsigned **__cmp_cat_id**
- template<> constexpr unsigned **__cmp_cat_id**< **partial_ordering** >
- template<> constexpr unsigned **__cmp_cat_id**< **strong_ordering** >
- template<> constexpr unsigned **__cmp_cat_id**< **weak_ordering** >
- template<typename _Tp>
  constexpr bool **__destructible**
- template<typename _Tp>
  constexpr bool **__destructible**< **_Tp &** >
- template<typename _Tp>
  constexpr bool **__destructible**< **_Tp &&** >
- template<typename _Tp, size_t _Nm>
  constexpr bool **__destructible**< **_Tp[_Nm]**>
- template<typename _Tp>
  constexpr bool **__destructible_impl**
- template<typename _Tp>
  constexpr bool **__destructible_impl**< **_Tp** >
- template<typename _Tp>
  constexpr bool **__is_ref_wrapper**
- template<typename _Tp>
  constexpr bool **__is_ref_wrapper**< **reference_wrapper**< **_Tp** > >
- constexpr _StateIdT **_S_invalid_state_id**

4.12.1 Detailed Description

Implementation details not part of the namespace std interface.

4.12.2 Function Documentation

__from_chars_alnum()

```
template<bool _DecOnly, typename _Tp>
bool std::__detail::__from_chars_alnum (
             const char *& __first,
             const char * __last,
             _Tp & __val,
             int __base)  [constexpr]
```

std::from_chars implementation for integers in any base. If _DecOnly is true, then we may assume __base is at most 10.
Referenced by std::from_chars().

__from_chars_pow2_base()

```
template<bool _DecOnly, typename _Tp>
bool std::__detail::__from_chars_pow2_base (
```

```
            const char *& __first,
            const char * __last,
            _Tp & __val,
            int __base)  [constexpr]
```

std::from_chars implementation for integers in a power-of-two base. If _DecOnly is true, then we may assume __base is at most 8.
Referenced by std::from_chars().

operator<<() [1/2]

```
template<typename _CharT, typename _Traits, typename _String>
std::basic_ostream< _CharT, _Traits > & std::__detail::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const _Quoted_string< _String, _CharT > & __str)
```

Inserter for quoted strings.

operator<<() [2/2]

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & std::__detail::operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const _Quoted_string< const _CharT *, _CharT > & __str)
```

Inserter for quoted strings.

operator>>()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istream< _CharT, _Traits > & std::__detail::operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            const _Quoted_string< basic_string< _CharT, _Traits, _Alloc > &, _CharT > & __str)
```

Extractor for delimited strings. The left and right delimiters can be different.
References std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::flags(), std::basic_ios< _CharT, _Traits >::good(), std::ios_base::setf(), std::ios_base::skipws, and std::basic_istream< _CharT, _Traits >::unget().

4.13 std::__parallel Namespace Reference

Classes

- struct _CRandNumber

Functions

- template<typename __RAIter, typename _Tp, typename _BinaryOperation>
  _Tp **__accumulate_switch** (__RAIter __begin, __RAIter __end, _Tp __init, _BinaryOperation __binary_op, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOperation, typename _IteratorTag>
  _Tp **__accumulate_switch** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)
- template<typename _IIter, typename _Tp, typename _IteratorTag>
  _Tp **__accumulate_switch** (_IIter __begin, _IIter __end, _Tp __init, _IteratorTag)
- template<typename _IIter, typename _Tp, typename _BinaryOper, typename _Tag>
  _Tp **__accumulate_switch** (_IIter, _IIter, _Tp, _BinaryOper, _Tag)
- template<typename _IIter, typename _Tp, typename _Tag>
  _Tp **__accumulate_switch** (_IIter, _IIter, _Tp, _Tag)

- template<typename _RAIter, typename _Tp, typename _BinaryOper>
  _Tp **__accumulate_switch** (_RAIter, _RAIter, _Tp, _BinaryOper, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>
  _OutputIterator **__adjacent_difference_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _Binary↩ Operation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **__adjacent_difference_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _Binary↩ Operation __bin_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism ↩ __parallelism_tag)
- template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>
  _OIter **__adjacent_difference_switch** (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **__adjacent_difference_switch** (_IIter, _IIter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _FIter, typename _BiPredicate, typename _IterTag>
  _FIter **__adjacent_find_switch** (_FIter, _FIter, _BiPredicate, _IterTag)
- template<typename _FIter, typename _IterTag>
  _FIter **__adjacent_find_switch** (_FIter, _FIter, _IterTag)
- template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag>
  _FIterator **__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred, _Iterator↩ Tag)
- template<typename _FIterator, typename _IteratorTag>
  _FIterator **__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _IteratorTag)
- template<typename _RAIter, typename _BinaryPredicate>
  _RAIter **__adjacent_find_switch** (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random_access_iterator_tag)
- template<typename _RAIter>
  _RAIter **__adjacent_find_switch** (_RAIter __begin, _RAIter __end, random_access_iterator_tag)
- template<typename _RAIter, typename _BiPredicate>
  _RAIter **__adjacent_find_switch** (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)
- template<typename _IIter, typename _Predicate, typename _IteratorTag>
  iterator_traits< _IIter >::difference_type **__count_if_switch** (_IIter __begin, _IIter __end, _Predicate __pred, _IteratorTag)
- template<typename _IIter, typename _Predicate, typename _IterTag>
  iterator_traits< _IIter >::difference_type **__count_if_switch** (_IIter, _IIter, _Predicate, _IterTag)
- template<typename _RAIter, typename _Predicate>
  iterator_traits< _RAIter >::difference_type **__count_if_switch** (_RAIter __begin, _RAIter __end, _Predicate _↩ _pred, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp, typename _IteratorTag>
  iterator_traits< _IIter >::difference_type **__count_switch** (_IIter __begin, _IIter __end, const _Tp &__value, _↩ IteratorTag)
- template<typename _IIter, typename _Tp, typename _IterTag>
  iterator_traits< _IIter >::difference_type **__count_switch** (_IIter, _IIter, const _Tp &, _IterTag)
- template<typename _RAIter, typename _Tp>
  iterator_traits< _RAIter >::difference_type **__count_switch** (_RAIter __begin, _RAIter __end, const _Tp &__↩ value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  bool **__equal_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  bool **__equal_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)

- template<typename _IIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>
  _IIter **__find_first_of_switch** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _Binary↩
  Predicate __comp, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2>
  _IIter **__find_first_of_switch** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _Iterator↩
  Tag1, _IteratorTag2)
- template<typename _IIter, typename _FIter, typename _BiPredicate, typename _IterTag1, typename _IterTag2>
  _IIter **__find_first_of_switch** (_IIter, _IIter, _FIter, _FIter, _BiPredicate, _IterTag1, _IterTag2)
- template<typename _IIter, typename _FIter, typename _IterTag1, typename _IterTag2>
  _IIter **__find_first_of_switch** (_IIter, _IIter, _FIter, _FIter, _IterTag1, _IterTag2)
- template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag>
  _RAIter **__find_first_of_switch** (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, random_access_iterator_tag, _IteratorTag)
- template<typename _RAIter, typename _FIter, typename _BiPredicate, typename _IterTag>
  _RAIter **__find_first_of_switch** (_RAIter, _RAIter, _FIter, _FIter, _BiPredicate, random_access_iterator_tag, _↩
  IterTag)
- template<typename _IIter, typename _Predicate, typename _IteratorTag>
  _IIter **__find_if_switch** (_IIter __begin, _IIter __end, _Predicate __pred, _IteratorTag)
- template<typename _IIter, typename _Predicate, typename _IterTag>
  _IIter **__find_if_switch** (_IIter, _IIter, _Predicate, _IterTag)
- template<typename _RAIter, typename _Predicate>
  _RAIter **__find_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _IIter, typename _Tp, typename _IteratorTag>
  _IIter **__find_switch** (_IIter __begin, _IIter __end, const _Tp &__val, _IteratorTag)
- template<typename _IIter, typename _Tp, typename _IterTag>
  _IIter **__find_switch** (_IIter, _IIter, const _Tp &, _IterTag)
- template<typename _RAIter, typename _Tp>
  _RAIter **__find_switch** (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)
- template<typename _IIter, typename _Function, typename _IteratorTag>
  _Function **__for_each_switch** (_IIter __begin, _IIter __end, _Function __f, _IteratorTag)
- template<typename _IIter, typename _Function, typename _IterTag>
  _Function **__for_each_switch** (_IIter, _IIter, _Function, _IterTag)
- template<typename _RAIter, typename _Function>
  _Function **__for_each_switch** (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _OIter, typename _Size, typename _Generator, typename _IterTag>
  _OIter **__generate_n_switch** (_OIter, _Size, _Generator, _IterTag)
- template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag>
  _OutputIterator **__generate_n_switch** (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)
- template<typename _RAIter, typename _Size, typename _Generator>
  _RAIter **__generate_n_switch** (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Generator, typename _IterTag>
  void **__generate_switch** (_FIter, _FIter, _Generator, _IterTag)
- template<typename _FIterator, typename _Generator, typename _IteratorTag>
  void **__generate_switch** (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)
- template<typename _RAIter, typename _Generator>
  void **__generate_switch** (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _↩
  IteratorTag1, typename _IteratorTag2>
  _Tp **__inner_product_switch** (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init, _BinaryFunction1 _↩
  _binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)

- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2>
  _Tp **__inner_product_switch** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, _Tag1, _Tag2)
- template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **__inner_product_switch** (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  bool **__lexicographical_compare_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, typename _IterTag2>
  bool **__lexicographical_compare_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _Predicate, _IterTag1, _IterTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  bool **__lexicographical_compare_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter, typename _Compare, typename _IterTag>
  _FIter **__max_element_switch** (_FIter, _FIter, _Compare, _IterTag)
- template<typename _FIterator, typename _Compare, typename _IteratorTag>
  _FIterator **__max_element_switch** (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)
- template<typename _RAIter, typename _Compare>
  _RAIter **__max_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>
  _OutputIterator **__merge_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare>
  _OutputIterator **__merge_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **__merge_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  _OIter **__merge_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter, typename _Compare, typename _IterTag>
  _FIter **__min_element_switch** (_FIter, _FIter, _Compare, _IterTag)
- template<typename _FIterator, typename _Compare, typename _IteratorTag>
  _FIterator **__min_element_switch** (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)
- template<typename _RAIter, typename _Compare>
  _RAIter **__min_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  pair< _IIter1, _IIter2 > **__mismatch_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  pair< _IIter1, _IIter2 > **__mismatch_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate _pred, _IteratorTag1, _IteratorTag2)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, typename _IterTag2>
  pair< _IIter1, _IIter2 > **__mismatch_switch** (_IIter1, _IIter1, _IIter2, _Predicate, _IterTag1, _IterTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  pair< _RAIter1, _RAIter2 > **__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)

- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  pair< _RAIter1, _RAIter2 > **__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>
  _OutputIterator **__partial_sum_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **__partial_sum_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>
  _OIter **__partial_sum_switch** (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **__partial_sum_switch** (_IIter, _IIter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter, typename _Predicate, typename _IterTag>
  _FIter **__partition_switch** (_FIter, _FIter, _Predicate, _IterTag)
- template<typename _FIterator, typename _Predicate, typename _IteratorTag>
  _FIterator **__partition_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, _IteratorTag)
- template<typename _RAIter, typename _Predicate>
  _RAIter **__partition_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _FIter, typename _Predicate, typename _Tp, typename _IterTag>
  void **__replace_if_switch** (_FIter, _FIter, _Predicate, const _Tp &, _IterTag)
- template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag>
  void **__replace_if_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)
- template<typename _RAIter, typename _Predicate, typename _Tp>
  void **__replace_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Tp, typename _IterTag>
  void **__replace_switch** (_FIter, _FIter, const _Tp &, const _Tp &, _IterTag)
- template<typename _FIterator, typename _Tp, typename _IteratorTag>
  void **__replace_switch** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)
- template<typename _RAIter, typename _Tp>
  void **__replace_switch** (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag>
  _FIter **__search_n_switch** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, _IterTag)
- template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag>
  _FIterator **__search_n_switch** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)
- template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate>
  _RAIter **__search_n_switch** (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, random_access_iterator_tag)
- template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate>
  _RAIter **__search_n_switch** (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)
- template<typename _FIter1, typename _FIter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2>
  _FIter1 **__search_switch** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, _IterTag1, _IterTag2)
- template<typename _FIter1, typename _FIter2, typename _IterTag1, typename _IterTag2>
  _FIter1 **__search_switch** (_FIter1, _FIter1, _FIter2, _FIter2, _IterTag1, _IterTag2)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>
  _FIterator1 **__search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)

- template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2>
  _FIterator1 **__search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __↩ end2, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate>
  _RAIter1 **__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _↩ BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter1, typename _RAIter2>
  _RAIter1 **__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter1, typename _RAIter2, typename _BiPredicate>
  _RAIter1 **__search_switch** (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _↩ IteratorTag2, typename _IteratorTag3>
  _OutputIterator **__set_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **__set_difference_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _Iter↩ Tag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **__set_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _↩ IteratorTag2, typename _IteratorTag3>
  _OutputIterator **__set_intersection_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **__set_intersection_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _↩ IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **__set_intersection_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _↩ IteratorTag2, typename _IteratorTag3>
  _OutputIterator **__set_symmetric_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **__set_symmetric_difference_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _↩ IterTag2, _IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **__set_symmetric_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 ↩ __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _↩ IteratorTag2, typename _IteratorTag3>
  _OutputIterator **__set_union_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _↩ OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)

- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **__set_union_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **__set_union_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2>
  _OIter **__transform1_switch** (_IIter, _IIter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)
- template<typename _RAIIter, typename _RAOIter, typename _UnaryOperation>
  _RAOIter **__transform1_switch** (_RAIIter, _RAIIter, _RAOIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_balanced)
- template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2>
  _RAIter2 **__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation>
  _RAIter2 **__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3>
  _OutputIterator **__transform2_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _Tag3>
  _OIter **__transform2_switch** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)
- template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation>
  _RAIter3 **__transform2_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation>
  _RAIter3 **__transform2_switch** (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_bal
- template<typename _IIter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  _OutputIterator **__unique_copy_switch** (_IIter __begin, _IIter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2>
  _OIter **__unique_copy_switch** (_IIter, _IIter, _OIter, _Predicate, _IterTag1, _IterTag2)
- template<typename _RAIter, typename _RandomAccessOutputIterator, typename _Predicate>
  _RandomAccessOutputIterator **__unique_copy_switch** (_RAIter __begin, _RAIter __last, _RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate>
  _RandomAccess_OIter **__unique_copy_switch** (_RAIter, _RAIter, _RandomAccess_OIter, _Predicate, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOperation>
  _Tp **accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op)
- template<typename _IIter, typename _Tp, typename _BinaryOperation>
  _Tp **accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOperation>
  _Tp **accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp>
  _Tp **accumulate** (_IIter, _IIter, _Tp)

- template<typename _IIter, typename _Tp>
  _Tp **accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _Tp>
  _Tp **accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOper>
  _Tp **accumulate** (_IIter, _IIter, _Tp, _BinaryOper)
- template<typename _IIter, typename _Tp, typename _BinaryOper>
  _Tp **accumulate** (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _Tp, typename _BinaryOper>
  _Tp **accumulate** (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _OIter>
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter)
- template<typename _IIter, typename _OIter>
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _OIter>
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _FIter>
  _FIter **adjacent_find** (_FIter, _FIter)
- template<typename _FIter>
  _FIter **adjacent_find** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _BiPredicate>
  _FIter **adjacent_find** (_FIter, _FIter, _BiPredicate)
- template<typename _FIter, typename _BiPredicate>
  _FIter **adjacent_find** (_FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _FIterator>
  _FIterator **adjacent_find** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator>
  _FIterator **adjacent_find** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _BinaryPredicate>
  _FIterator **adjacent_find** (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)

- template<typename _FIterator, typename _BinaryPredicate>
  _FIterator **adjacent_find** (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **count** (_IIter __begin, _IIter __end, const _Tp &__value)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **count_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1, typename _IIter2>
  bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template<typename _IIter1, typename _IIter2>
  bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _BinaryPredicate __binary_pred)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  constexpr bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  bool **equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp>
  _IIter **find** (_IIter __begin, _IIter __end, const _Tp &__val)
- template<typename _IIter, typename _Tp>
  _IIter **find** (_IIter __begin, _IIter __end, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIterator>
  _IIter **find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2)
- template<typename _IIter, typename _FIterator>
  _IIter **find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIterator, typename _BinaryPredicate>
  _IIter **find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp)
- template<typename _IIter, typename _FIterator, typename _BinaryPredicate>
  _IIter **find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIter>
  _IIter **find_first_of** (_IIter, _IIter, _FIter, _FIter)

- template<typename _IIter, typename _FIter>
  _IIter **find_first_of** (_IIter, _IIter, _FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIter, typename _BiPredicate>
  _IIter **find_first_of** (_IIter, _IIter, _FIter, _FIter, _BiPredicate)
- template<typename _IIter, typename _FIter, typename _BiPredicate>
  _IIter **find_first_of** (_IIter, _IIter, _FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Predicate>
  _IIter **find_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  _IIter **find_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Function>
  _Function **for_each** (_IIter __begin, _IIter __end, _Function __f, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Function>
  _Function **for_each** (_IIter, _IIter, _Function)
- template<typename _Iterator, typename _Function>
  _Function **for_each** (_Iterator __begin, _Iterator __end, _Function __f)
- template<typename _Iterator, typename _Function>
  _Function **for_each** (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::_Parallelism __↩
  parallelism_tag)
- template<typename _FIter, typename _Generator>
  void **generate** (_FIter, _FIter, _Generator)
- template<typename _FIter, typename _Generator>
  void **generate** (_FIter, _FIter, _Generator, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Generator>
  void **generate** (_FIter, _FIter, _Generator, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Generator>
  void **generate** (_FIterator __begin, _FIterator __end, _Generator __gen)
- template<typename _FIterator, typename _Generator>
  void **generate** (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::_Parallelism __↩
  parallelism_tag)
- template<typename _FIterator, typename _Generator>
  void **generate** (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::sequential_tag)
- template<typename _OIter, typename _Size, typename _Generator>
  _OIter **generate_n** (_OIter, _Size, _Generator)
- template<typename _OIter, typename _Size, typename _Generator>
  _OIter **generate_n** (_OIter, _Size, _Generator, __gnu_parallel::_Parallelism)
- template<typename _OIter, typename _Size, typename _Generator>
  _OIter **generate_n** (_OIter, _Size, _Generator, __gnu_parallel::sequential_tag)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  _OutputIterator **generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  _OutputIterator **generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::_Parallelism
  __parallelism_tag)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  _OutputIterator **generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2)

- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template<typename _IIter1, typename _IIter2>
  bool **lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  constexpr bool **lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  bool **lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _FIter>
  _FIter **max_element** (_FIter, _FIter)
- template<typename _FIter>
  _FIter **max_element** (_FIter, _FIter, __gnu_parallel::_Parallelism)
- template<typename _FIter>
  _FIter **max_element** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Compare>
  _FIter **max_element** (_FIter, _FIter, _Compare)
- template<typename _FIter, typename _Compare>
  _FIter **max_element** (_FIter, _FIter, _Compare, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Compare>
  _FIter **max_element** (_FIter, _FIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _FIterator>
  _FIterator **max_element** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator>
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator>
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, _Compare __comp)
- template<typename _FIterator, typename _Compare>
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **max_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare>
  _OutputIterator **merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare>
  _OutputIterator **merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, __gnu_parallel::sequential_tag)

- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  _OIter **merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _FIter>
  _FIter **min_element** (_FIter, _FIter)
- template<typename _FIter>
  _FIter **min_element** (_FIter, _FIter, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter>
  _FIter **min_element** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Compare>
  _FIter **min_element** (_FIter, _FIter, _Compare)
- template<typename _FIter, typename _Compare>
  _FIter **min_element** (_FIter, _FIter, _Compare, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Compare>
  _FIter **min_element** (_FIter, _FIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _FIterator>
  _FIterator **min_element** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator>
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator>
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, _Compare __comp)
- template<typename _FIterator, typename _Compare>
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **min_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  pair< _IIter1, _IIter2 > **mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  pair< _InputIterator1, _InputIterator2 > **mismatch** (_InputIterator1 __begin1, _InputIterator1 __end1, _Input↩
  Iterator2 __begin2, _InputIterator2 __end2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2>
  pair< _InputIterator1, _InputIterator2 > **mismatch** (_InputIterator1 __first1, _InputIterator1 __last1, _Input↩
  Iterator2 __first2, _InputIterator2 __last2, __gnu_parallel::sequential_tag)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  pair< _InputIterator1, _InputIterator2 > **mismatch** (_InputIterator1 __first1, _InputIterator1 __last1, _Input↩
  Iterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)

- template<typename _RAIter>
  void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end)
- template<typename _RAIter>
  void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end)
- template<typename _RAIter>
  void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __binary↩
  _op)
- template<typename _IIter, typename _OIter>
  _OIter **partial_sum** (_IIter, _IIter, _OIter __result)
- template<typename _IIter, typename _OIter>
  _OIter **partial_sum** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **partial_sum** (_IIter, _IIter, _OIter, _BinaryOper)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **partial_sum** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Predicate>
  _FIter **partition** (_FIter, _FIter, _Predicate)
- template<typename _FIter, typename _Predicate>
  _FIter **partition** (_FIter, _FIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Predicate>
  _FIterator **partition** (_FIterator __begin, _FIterator __end, _Predicate __pred)
- template<typename _FIterator, typename _Predicate>
  _FIterator **partition** (_FIterator __begin, _FIterator __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **random_shuffle** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **random_shuffle** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void **random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void **random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Tp>
  void **replace** (_FIter, _FIter, const _Tp &, const _Tp &)

- template<typename _FIter, typename _Tp>
  void **replace** (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Tp>
  void **replace** (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Tp>
  void **replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _FIterator, typename _Tp>
  void **replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Tp>
  void **replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Predicate, typename _Tp>
  void **replace_if** (_FIter, _FIter, _Predicate, const _Tp &)
- template<typename _FIter, typename _Predicate, typename _Tp>
  void **replace_if** (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Predicate, typename _Tp>
  void **replace_if** (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Predicate, typename _Tp>
  void **replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value)
- template<typename _FIterator, typename _Predicate, typename _Tp>
  void **replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Predicate, typename _Tp>
  void **replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, __gnu_parallel::sequential_tag)
- template<typename _FIter1, typename _FIter2>
  _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2>
  _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2, __gnu_parallel::sequential_tag)
- template<typename _FIter1, typename _FIter2, typename _BiPredicate>
  constexpr _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate)
- template<typename _FIter1, typename _FIter2, typename _BiPredicate>
  _FIter1 **search** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _FIterator1, typename _FIterator2>
  _FIterator1 **search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)
- template<typename _FIterator1, typename _FIterator2>
  _FIterator1 **search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, __gnu_parallel::sequential_tag)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate>
  constexpr _FIterator1 **search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __↩
  end2, _BinaryPredicate __pred)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate>
  _FIterator1 **search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _↩
  BinaryPredicate __pred, __gnu_parallel::sequential_tag)
- template<typename _ForwardIterator, typename _Searcher>
  _ForwardIterator search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)
- template<typename _FIter, typename _Integer, typename _Tp>
  _FIter **search_n** (_FIter, _FIter, _Integer, const _Tp &)
- template<typename _FIter, typename _Integer, typename _Tp>
  _FIter **search_n** (_FIter, _FIter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate>
  _FIter **search_n** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate)

- template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate>
  _FIter **search_n** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Integer, typename _Tp>
  _FIterator **search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)
- template<typename _FIterator, typename _Integer, typename _Tp>
  _FIterator **search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
  _FIterator **search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)
- template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
  _FIterator **search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↩Iterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↩Iterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↩Iterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↩Iterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↩Iterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↩Iterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↩Iterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Output↩Iterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)

- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_exact_tag __parallelism)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_sampling_tag __parallelism)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)

- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)
- template<typename _RAIter>
  void **sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare, typename _Parallelism>
  void **sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)
- template<typename _RAIter>
  void **stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare, typename _Parallelism>
  void **stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)
- template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>
  _OutputIterator **transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _UnaryOperation __unary_op)
- template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>
  _OutputIterator **transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>
  _OutputIterator **transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _UnaryOperation __unary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  _OIter **transform** (_IIter, _IIter, _OIter, _UnaryOperation)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  _OIter **transform** (_IIter, _IIter, _OIter, _UnaryOperation, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  _OIter **transform** (_IIter, _IIter, _OIter, _UnaryOperation, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _↩
  BinaryOperation __binary_op)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _↩
  BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)

- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _↩
  BinaryOperation __binary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation>
  _OIter **transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation>
  _OIter **transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation>
  _OIter **transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _Predicate>
  _OutputIterator **unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter, typename _OutputIterator, typename _Predicate>
  _OutputIterator **unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter>
  _OIter **unique_copy** (_IIter, _IIter, _OIter)
- template<typename _IIter, typename _OIter>
  _OIter **unique_copy** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _Predicate>
  _OIter **unique_copy** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _IIter, typename _OIter, typename _Predicate>
  _OIter **unique_copy** (_IIter, _IIter, _OIter, _Predicate, __gnu_parallel::sequential_tag)

4.13.1 Detailed Description

GNU parallel code, replaces standard behavior with parallel behavior.

4.13.2 Function Documentation

search()

```
template<typename _ForwardIterator, typename _Searcher>
_ForwardIterator std::__parallel::search (
            _ForwardIterator __first,
            _ForwardIterator __last,
            const _Searcher & __searcher)  [inline]
```

Search a sequence using a Searcher object.

Parameters

__first	A forward iterator.
__last	A forward iterator.
__searcher	A callable object.

Returns

`__searcher(__first,__last)`.first

4.14 std::chrono Namespace Reference

Classes

- class duration
- struct duration_values
- class gps_clock
- class hh_mm_ss
- struct steady_clock
- struct system_clock
- class tai_clock
- class time_point
- struct treat_as_floating_point
- class tzdb_list
- class utc_clock

Typedefs

- using days
- using **file_clock**
- template<typename _Duration>
  using **file_time**
- using **gps_seconds**
- template<typename _Duration>
  using **gps_time**
- using high_resolution_clock
- using hours
- using **local_days**
- using **local_seconds**
- template<typename _Duration>
  using **local_time**
- using microseconds
- using milliseconds
- using minutes
- using months
- using nanoseconds
- using seconds
- using **sys_days**
- using **sys_seconds**
- template<typename _Duration>
  using **sys_time**
- using **tai_seconds**
- template<typename _Duration>
  using **tai_time**
- using **utc_seconds**
- template<typename _Duration>
  using **utc_time**
- using weeks
- using years
- using **zoned_seconds**

Enumerations

- enum class **choose** { **earliest** , **latest** }

Functions

- template<typename _Rep, typename _Period>
  constexpr enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > > abs (duration< _Rep, _Period > __d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > ceil (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > ceil (const time_point< _Clock, _Dur > &__tp)
- template<typename _DestClock, typename _SourceClock, typename _Duration>
  requires __detail::__clock_convs<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_utc<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys_utc<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_utc_sys<_DestClock, _SourceClock, _Duration>
  auto clock_cast (const time_point< _SourceClock, _Duration > &__t)
- const time_zone * **current_zone** ()
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > duration_cast (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > floor (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > floor (const time_point< _Clock, _Dur > &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, day &__d, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Rep, typename _Period, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, duration< _Rep, _Period > &__d, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, file_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, gps_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *_offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, local_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *_offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, month &__m, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
  basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *_fmt, month_day &__md, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)

- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
__fmt, sys_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__↩
offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
__fmt, tai_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__↩
offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
__fmt, utc_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__↩
offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
__fmt, weekday &__wd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *↩
__fmt, year &__y, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *_↩
_fmt, year_month &__ym, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT
*__fmt, year_month_day &__ymd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__↩
offset=nullptr)
- template<typename _Duration>
leap_second_info **get_leap_second_info** (const utc_time< _Duration > &__ut)
- const tzdb & **get_tzdb** ()
- tzdb_list & **get_tzdb_list** ()
- constexpr bool is_am (const hours &__h) noexcept
- constexpr bool is_pm (const hours &__h) noexcept
- template<typename _Duration>
__detail::__local_time_fmt< _Duration > local_time_format (local_time< _Duration > __time, const string *_↩
_abbrev=nullptr, const seconds *__offset_sec=nullptr)
- const time_zone * **locate_zone** (string_view __tz_name)
- constexpr hours make12 (const hours &__h) noexcept
- constexpr hours make24 (const hours &__h, bool __is_pm) noexcept
- template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const day &__d)
- template<typename _CharT, typename _Traits, typename _Duration>
basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const file_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const gps_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const hh_mm_ss< _Duration > &__hms)
- template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const local_info &__li)

- template<typename _CharT, typename _Traits, typename _Duration>
  requires requires(const sys_time<_Duration>& __st) { __os << __st; }
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const local_time< _Duration > &__lt)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const month &__m)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const month_day &__md)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const month_day_last &__mdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const month_weekday &__mwd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const month_weekday_last &__mwdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const sys_days &__dp)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const sys_info &__i)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const sys_time< _Duration > &__tp)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const tai_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const utc_time< _Duration > &__t)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const weekday &__wd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const weekday_indexed &__wdi)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const weekday_last &__wdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const year &__y)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const year_month &__ym)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const year_month_day &__ymd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const year_month_day_last &__ymdl)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const year_↩
  month_weekday &__ymwd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const year_↩
  month_weekday_last &__ymwdl)
- template<typename _CharT, typename _Traits, typename _Duration, typename _TimeZonePtr>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__os, const zoned_↩
  time< _Duration, _TimeZonePtr > &__t)
- template<typename _CharT, typename _Traits, typename _Rep, typename _Period>
  basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const
  duration< _Rep, _Period > &__d)
- template<typename _Dur1, typename _TZPtr1, typename _Dur2, typename _TZPtr2>
  bool **operator==** (const zoned_time< _Dur1, _TZPtr1 > &__x, const zoned_time< _Dur2, _TZPtr2 > &__y)
- template<typename _CharT, __detail::__parsable< _CharT > _Parsable>
  auto **parse** (const _CharT *__fmt, _Parsable &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__↩
  parsable< _CharT, _Traits, _StrT > _Parsable>
  auto **parse** (const _CharT *__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__↩
  parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **parse** (const _CharT *__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev, minutes
  &__offset)
- template<typename _CharT, typename _Traits = char_traits<_CharT>, typename _StrT = basic_string<_CharT, _Traits>, __detail::__↩
  parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **parse** (const _CharT *__fmt, _Parsable &__tp, minutes &__offset)
- template<typename _CharT, typename _Traits, typename _Alloc, __detail::__parsable< _CharT, _Traits > _Parsable>
  auto **parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__↩
  parsable< _CharT, _Traits, _StrT > _Parsable>
  auto **parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, _↩
  Traits, _Alloc > &__abbrev)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__↩
  parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, _↩
  Traits, _Alloc > &__abbrev, minutes &__offset)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable<
  _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, minutes &__offset)
- const tzdb & **reload_tzdb** ()
- string **remote_version** ()
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr enable_if_t< __and_< __is_duration< _ToDur >, __not_< treat_as_floating_point< typename _To↩
  Dur::rep > > >::value, _ToDur > round (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur > &&!treat_as_floating_point_v< typename _ToDur::rep >,
  time_point< _Clock, _ToDur > > round (const time_point< _Clock, _Dur > &__tp)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr __enable_if_t< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > > time_point_cast
  (const time_point< _Clock, _Dur > &__t)
- **zoned_time** () -> zoned_time< seconds >
- template<typename _TimeZonePtrOrName>
  zoned_time (_TimeZonePtrOrName &&) -> zoned_time< seconds, __time_zone_representation< _Time↩
  ZonePtrOrName > >

- template<typename _TimeZonePtrOrName, typename _Duration>
  zoned_time (_TimeZonePtrOrName &&, local_time< _Duration >, choose=choose::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _TimeZonePtrOrName, typename _Duration>
  zoned_time (_TimeZonePtrOrName &&, sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _Duration, typename _TimeZonePtrOrName, typename _TimeZonePtr2>
  zoned_time (_TimeZonePtrOrName &&, zoned_time< _Duration, _TimeZonePtr2 >, choose=choose::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _Duration>
  zoned_time (sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds > >

- template<typename _Clock, typename _Dur1, three_way_comparable_with< _Dur1 > _Dur2>
  constexpr auto **operator<=>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **operator<** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **operator<=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **operator>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **operator>=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)

Variables

- constexpr month **April**
- constexpr month **August**
- constexpr month **December**
- constexpr month **February**
- constexpr weekday **Friday**
- template<typename _Tp>
  constexpr bool **is_clock_v**
- template<typename _Tp>
  constexpr bool **is_clock_v**< **_Tp** >
- template<> constexpr bool **is_clock_v**< **file_clock** >
- template<> constexpr bool **is_clock_v**< **gps_clock** >
- template<> constexpr bool **is_clock_v**< **steady_clock** >
- template<> constexpr bool **is_clock_v**< **system_clock** >
- template<> constexpr bool **is_clock_v**< **tai_clock** >
- template<> constexpr bool **is_clock_v**< **utc_clock** >
- constexpr month **January**
- constexpr month **July**
- constexpr month **June**
- constexpr last_spec **last**
- constexpr month **March**
- constexpr month **May**

- constexpr weekday **Monday**
- constexpr month **November**
- constexpr month **October**
- constexpr weekday **Saturday**
- constexpr month **September**
- constexpr weekday **Sunday**
- constexpr weekday **Thursday**
- template<typename _Rep>
  constexpr bool **treat_as_floating_point_v**
- template<> constexpr bool **treat_as_floating_point_v**< **double** >
- template<> constexpr bool **treat_as_floating_point_v**< **float** >
- template<> constexpr bool **treat_as_floating_point_v**< **int** >
- template<> constexpr bool **treat_as_floating_point_v**< **long** >
- template<> constexpr bool **treat_as_floating_point_v**< **long double** >
- template<> constexpr bool **treat_as_floating_point_v**< **long long** >
- constexpr weekday **Tuesday**
- constexpr weekday **Wednesday**

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator- (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Rep2, typename _Period>
  constexpr duration< __common_rep_t< _Rep2, _Rep1 >, _Period > operator* (const _Rep1 &__s, const duration< _Rep2, _Period > &__d)
- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > operator/ (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< _Rep1, _Rep2 >::type operator/ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > operator% (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator% (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (const duration< _Rep1, _↩
  Period > &__d, const _Rep2 &__s)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr auto operator<=> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>
  constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type > operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
  constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr common_type< _Dur1, _Dur2 >::type operator- (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
  constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator+ (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

4.14.1 Detailed Description

ISO C++ 2011 namespace for date and time utilities.

4.15 std::decimal Namespace Reference

Classes

- class decimal128
- class decimal32
- class decimal64

Functions

- double **decimal128_to_double** (decimal128 __d)
- float **decimal128_to_float** (decimal128 __d)
- long double **decimal128_to_long_double** (decimal128 __d)
- long long **decimal128_to_long_long** (decimal128 __d)
- double **decimal32_to_double** (decimal32 __d)
- float **decimal32_to_float** (decimal32 __d)
- long double **decimal32_to_long_double** (decimal32 __d)
- long long decimal32_to_long_long (decimal32 __d)
- double **decimal64_to_double** (decimal64 __d)
- float **decimal64_to_float** (decimal64 __d)
- long double **decimal64_to_long_double** (decimal64 __d)
- long long **decimal64_to_long_long** (decimal64 __d)
- double **decimal_to_double** (decimal128 __d)
- double **decimal_to_double** (decimal32 __d)
- double **decimal_to_double** (decimal64 __d)
- float **decimal_to_float** (decimal128 __d)

- float **decimal_to_float** (decimal32 __d)
- float **decimal_to_float** (decimal64 __d)
- long double **decimal_to_long_double** (decimal128 __d)
- long double **decimal_to_long_double** (decimal32 __d)
- long double **decimal_to_long_double** (decimal64 __d)
- long long **decimal_to_long_long** (decimal128 __d)
- long long **decimal_to_long_long** (decimal32 __d)
- long long **decimal_to_long_long** (decimal64 __d)
- static decimal128 **make_decimal128** (long long __coeff, int __exp)
- static decimal128 **make_decimal128** (unsigned long long __coeff, int __exp)
- static decimal32 **make_decimal32** (long long __coeff, int __exp)
- static decimal32 **make_decimal32** (unsigned long long __coeff, int __exp)
- static decimal64 **make_decimal64** (long long __coeff, int __exp)
- static decimal64 **make_decimal64** (unsigned long long __coeff, int __exp)
- bool **operator!=** (decimal128 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal128 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal128 __lhs, int __rhs)
- bool **operator!=** (decimal128 __lhs, long __rhs)
- bool **operator!=** (decimal128 __lhs, long long __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned int __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned long __rhs)
- bool **operator!=** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator!=** (decimal32 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal32 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal32 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal32 __lhs, int __rhs)
- bool **operator!=** (decimal32 __lhs, long __rhs)
- bool **operator!=** (decimal32 __lhs, long long __rhs)
- bool **operator!=** (decimal32 __lhs, unsigned int __rhs)
- bool **operator!=** (decimal32 __lhs, unsigned long __rhs)
- bool **operator!=** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator!=** (decimal64 __lhs, decimal128 __rhs)
- bool **operator!=** (decimal64 __lhs, decimal32 __rhs)
- bool **operator!=** (decimal64 __lhs, decimal64 __rhs)
- bool **operator!=** (decimal64 __lhs, int __rhs)
- bool **operator!=** (decimal64 __lhs, long __rhs)
- bool **operator!=** (decimal64 __lhs, long long __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned int __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned long __rhs)
- bool **operator!=** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator!=** (int __lhs, decimal128 __rhs)
- bool **operator!=** (int __lhs, decimal32 __rhs)
- bool **operator!=** (int __lhs, decimal64 __rhs)
- bool **operator!=** (long __lhs, decimal128 __rhs)
- bool **operator!=** (long __lhs, decimal32 __rhs)
- bool **operator!=** (long __lhs, decimal64 __rhs)
- bool **operator!=** (long long __lhs, decimal128 __rhs)
- bool **operator!=** (long long __lhs, decimal32 __rhs)
- bool **operator!=** (long long __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal128 __rhs)

- bool **operator!=** (unsigned int __lhs, decimal32 __rhs)
- bool **operator!=** (unsigned int __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal32 __rhs)
- bool **operator!=** (unsigned long __lhs, decimal64 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal128 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator!=** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **operator*** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **operator*** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **operator*** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **operator*** (decimal128 __lhs, int __rhs)
- decimal128 **operator*** (decimal128 __lhs, long __rhs)
- decimal128 **operator*** (decimal128 __lhs, long long __rhs)
- decimal128 **operator*** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **operator*** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **operator*** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **operator*** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **operator*** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **operator*** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **operator*** (decimal32 __lhs, int __rhs)
- decimal32 **operator*** (decimal32 __lhs, long __rhs)
- decimal32 **operator*** (decimal32 __lhs, long long __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator*** (decimal32 __lhs, unsigned long long __rhs)
- decimal128 **operator*** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **operator*** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **operator*** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **operator*** (decimal64 __lhs, int __rhs)
- decimal64 **operator*** (decimal64 __lhs, long __rhs)
- decimal64 **operator*** (decimal64 __lhs, long long __rhs)
- decimal64 **operator*** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **operator*** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **operator*** (decimal64 __lhs, unsigned long long __rhs)
- decimal128 **operator*** (int __lhs, decimal128 __rhs)
- decimal32 **operator*** (int __lhs, decimal32 __rhs)
- decimal64 **operator*** (int __lhs, decimal64 __rhs)
- decimal128 **operator*** (long __lhs, decimal128 __rhs)
- decimal32 **operator*** (long __lhs, decimal32 __rhs)
- decimal64 **operator*** (long __lhs, decimal64 __rhs)
- decimal128 **operator*** (long long __lhs, decimal128 __rhs)
- decimal32 **operator*** (long long __lhs, decimal32 __rhs)
- decimal64 **operator*** (long long __lhs, decimal64 __rhs)
- decimal128 **operator*** (unsigned int __lhs, decimal128 __rhs)
- decimal32 **operator*** (unsigned int __lhs, decimal32 __rhs)
- decimal64 **operator*** (unsigned int __lhs, decimal64 __rhs)
- decimal128 **operator*** (unsigned long __lhs, decimal128 __rhs)
- decimal32 **operator*** (unsigned long __lhs, decimal32 __rhs)
- decimal64 **operator*** (unsigned long __lhs, decimal64 __rhs)
- decimal128 **operator*** (unsigned long long __lhs, decimal128 __rhs)

- decimal32 **operator*** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **operator*** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **operator+** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **operator+** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **operator+** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **operator+** (decimal128 __lhs, int __rhs)
- decimal128 **operator+** (decimal128 __lhs, long __rhs)
- decimal128 **operator+** (decimal128 __lhs, long long __rhs)
- decimal128 **operator+** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **operator+** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **operator+** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **operator+** (decimal128 __rhs)
- decimal128 **operator+** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **operator+** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **operator+** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **operator+** (decimal32 __lhs, int __rhs)
- decimal32 **operator+** (decimal32 __lhs, long __rhs)
- decimal32 **operator+** (decimal32 __lhs, long long __rhs)
- decimal32 **operator+** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator+** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator+** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **operator+** (decimal32 __rhs)
- decimal128 **operator+** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **operator+** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **operator+** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **operator+** (decimal64 __lhs, int __rhs)
- decimal64 **operator+** (decimal64 __lhs, long __rhs)
- decimal64 **operator+** (decimal64 __lhs, long long __rhs)
- decimal64 **operator+** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **operator+** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **operator+** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **operator+** (decimal64 __rhs)
- decimal128 **operator+** (int __lhs, decimal128 __rhs)
- decimal32 **operator+** (int __lhs, decimal32 __rhs)
- decimal64 **operator+** (int __lhs, decimal64 __rhs)
- decimal128 **operator+** (long __lhs, decimal128 __rhs)
- decimal32 **operator+** (long __lhs, decimal32 __rhs)
- decimal64 **operator+** (long __lhs, decimal64 __rhs)
- decimal128 **operator+** (long long __lhs, decimal128 __rhs)
- decimal32 **operator+** (long long __lhs, decimal32 __rhs)
- decimal64 **operator+** (long long __lhs, decimal64 __rhs)
- decimal128 **operator+** (unsigned int __lhs, decimal128 __rhs)
- decimal32 **operator+** (unsigned int __lhs, decimal32 __rhs)
- decimal64 **operator+** (unsigned int __lhs, decimal64 __rhs)
- decimal128 **operator+** (unsigned long __lhs, decimal128 __rhs)
- decimal32 **operator+** (unsigned long __lhs, decimal32 __rhs)
- decimal64 **operator+** (unsigned long __lhs, decimal64 __rhs)
- decimal128 **operator+** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator+** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **operator+** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **operator-** (decimal128 __lhs, decimal128 __rhs)

- decimal128 **operator-** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **operator-** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **operator-** (decimal128 __lhs, int __rhs)
- decimal128 **operator-** (decimal128 __lhs, long __rhs)
- decimal128 **operator-** (decimal128 __lhs, long long __rhs)
- decimal128 **operator-** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **operator-** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **operator-** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **operator-** (decimal128 __rhs)
- decimal128 **operator-** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **operator-** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **operator-** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **operator-** (decimal32 __lhs, int __rhs)
- decimal32 **operator-** (decimal32 __lhs, long __rhs)
- decimal32 **operator-** (decimal32 __lhs, long long __rhs)
- decimal32 **operator-** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator-** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator-** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **operator-** (decimal32 __rhs)
- decimal128 **operator-** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **operator-** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **operator-** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **operator-** (decimal64 __lhs, int __rhs)
- decimal64 **operator-** (decimal64 __lhs, long __rhs)
- decimal64 **operator-** (decimal64 __lhs, long long __rhs)
- decimal64 **operator-** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **operator-** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **operator-** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **operator-** (decimal64 __rhs)
- decimal128 **operator-** (int __lhs, decimal128 __rhs)
- decimal32 **operator-** (int __lhs, decimal32 __rhs)
- decimal64 **operator-** (int __lhs, decimal64 __rhs)
- decimal128 **operator-** (long __lhs, decimal128 __rhs)
- decimal32 **operator-** (long __lhs, decimal32 __rhs)
- decimal64 **operator-** (long __lhs, decimal64 __rhs)
- decimal128 **operator-** (long long __lhs, decimal128 __rhs)
- decimal32 **operator-** (long long __lhs, decimal32 __rhs)
- decimal64 **operator-** (long long __lhs, decimal64 __rhs)
- decimal128 **operator-** (unsigned int __lhs, decimal128 __rhs)
- decimal32 **operator-** (unsigned int __lhs, decimal32 __rhs)
- decimal64 **operator-** (unsigned int __lhs, decimal64 __rhs)
- decimal128 **operator-** (unsigned long __lhs, decimal128 __rhs)
- decimal32 **operator-** (unsigned long __lhs, decimal32 __rhs)
- decimal64 **operator-** (unsigned long __lhs, decimal64 __rhs)
- decimal128 **operator-** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator-** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **operator-** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **operator/** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **operator/** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **operator/** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **operator/** (decimal128 __lhs, int __rhs)

- decimal128 **operator**/ (decimal128 __lhs, long __rhs)
- decimal128 **operator**/ (decimal128 __lhs, long long __rhs)
- decimal128 **operator**/ (decimal128 __lhs, unsigned int __rhs)
- decimal128 **operator**/ (decimal128 __lhs, unsigned long __rhs)
- decimal128 **operator**/ (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **operator**/ (decimal32 __lhs, decimal128 __rhs)
- decimal32 **operator**/ (decimal32 __lhs, decimal32 __rhs)
- decimal64 **operator**/ (decimal32 __lhs, decimal64 __rhs)
- decimal32 **operator**/ (decimal32 __lhs, int __rhs)
- decimal32 **operator**/ (decimal32 __lhs, long __rhs)
- decimal32 **operator**/ (decimal32 __lhs, long long __rhs)
- decimal32 **operator**/ (decimal32 __lhs, unsigned int __rhs)
- decimal32 **operator**/ (decimal32 __lhs, unsigned long __rhs)
- decimal32 **operator**/ (decimal32 __lhs, unsigned long long __rhs)
- decimal128 **operator**/ (decimal64 __lhs, decimal128 __rhs)
- decimal64 **operator**/ (decimal64 __lhs, decimal32 __rhs)
- decimal64 **operator**/ (decimal64 __lhs, decimal64 __rhs)
- decimal64 **operator**/ (decimal64 __lhs, int __rhs)
- decimal64 **operator**/ (decimal64 __lhs, long __rhs)
- decimal64 **operator**/ (decimal64 __lhs, long long __rhs)
- decimal64 **operator**/ (decimal64 __lhs, unsigned int __rhs)
- decimal64 **operator**/ (decimal64 __lhs, unsigned long __rhs)
- decimal64 **operator**/ (decimal64 __lhs, unsigned long long __rhs)
- decimal128 **operator**/ (int __lhs, decimal128 __rhs)
- decimal32 **operator**/ (int __lhs, decimal32 __rhs)
- decimal64 **operator**/ (int __lhs, decimal64 __rhs)
- decimal128 **operator**/ (long __lhs, decimal128 __rhs)
- decimal32 **operator**/ (long __lhs, decimal32 __rhs)
- decimal64 **operator**/ (long __lhs, decimal64 __rhs)
- decimal128 **operator**/ (long long __lhs, decimal128 __rhs)
- decimal32 **operator**/ (long long __lhs, decimal32 __rhs)
- decimal64 **operator**/ (long long __lhs, decimal64 __rhs)
- decimal128 **operator**/ (unsigned int __lhs, decimal128 __rhs)
- decimal32 **operator**/ (unsigned int __lhs, decimal32 __rhs)
- decimal64 **operator**/ (unsigned int __lhs, decimal64 __rhs)
- decimal128 **operator**/ (unsigned long __lhs, decimal128 __rhs)
- decimal32 **operator**/ (unsigned long __lhs, decimal32 __rhs)
- decimal64 **operator**/ (unsigned long __lhs, decimal64 __rhs)
- decimal128 **operator**/ (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **operator**/ (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **operator**/ (unsigned long long __lhs, decimal64 __rhs)
- bool **operator**< (decimal128 __lhs, decimal128 __rhs)
- bool **operator**< (decimal128 __lhs, decimal32 __rhs)
- bool **operator**< (decimal128 __lhs, decimal64 __rhs)
- bool **operator**< (decimal128 __lhs, int __rhs)
- bool **operator**< (decimal128 __lhs, long __rhs)
- bool **operator**< (decimal128 __lhs, long long __rhs)
- bool **operator**< (decimal128 __lhs, unsigned int __rhs)
- bool **operator**< (decimal128 __lhs, unsigned long __rhs)
- bool **operator**< (decimal128 __lhs, unsigned long long __rhs)
- bool **operator**< (decimal32 __lhs, decimal128 __rhs)

- bool **operator**< (decimal32 __lhs, decimal32 __rhs)
- bool **operator**< (decimal32 __lhs, decimal64 __rhs)
- bool **operator**< (decimal32 __lhs, int __rhs)
- bool **operator**< (decimal32 __lhs, long __rhs)
- bool **operator**< (decimal32 __lhs, long long __rhs)
- bool **operator**< (decimal32 __lhs, unsigned int __rhs)
- bool **operator**< (decimal32 __lhs, unsigned long __rhs)
- bool **operator**< (decimal32 __lhs, unsigned long long __rhs)
- bool **operator**< (decimal64 __lhs, decimal128 __rhs)
- bool **operator**< (decimal64 __lhs, decimal32 __rhs)
- bool **operator**< (decimal64 __lhs, decimal64 __rhs)
- bool **operator**< (decimal64 __lhs, int __rhs)
- bool **operator**< (decimal64 __lhs, long __rhs)
- bool **operator**< (decimal64 __lhs, long long __rhs)
- bool **operator**< (decimal64 __lhs, unsigned int __rhs)
- bool **operator**< (decimal64 __lhs, unsigned long __rhs)
- bool **operator**< (decimal64 __lhs, unsigned long long __rhs)
- bool **operator**< (int __lhs, decimal128 __rhs)
- bool **operator**< (int __lhs, decimal32 __rhs)
- bool **operator**< (int __lhs, decimal64 __rhs)
- bool **operator**< (long __lhs, decimal128 __rhs)
- bool **operator**< (long __lhs, decimal32 __rhs)
- bool **operator**< (long __lhs, decimal64 __rhs)
- bool **operator**< (long long __lhs, decimal128 __rhs)
- bool **operator**< (long long __lhs, decimal32 __rhs)
- bool **operator**< (long long __lhs, decimal64 __rhs)
- bool **operator**< (unsigned int __lhs, decimal128 __rhs)
- bool **operator**< (unsigned int __lhs, decimal32 __rhs)
- bool **operator**< (unsigned int __lhs, decimal64 __rhs)
- bool **operator**< (unsigned long __lhs, decimal128 __rhs)
- bool **operator**< (unsigned long __lhs, decimal32 __rhs)
- bool **operator**< (unsigned long __lhs, decimal64 __rhs)
- bool **operator**< (unsigned long long __lhs, decimal128 __rhs)
- bool **operator**< (unsigned long long __lhs, decimal32 __rhs)
- bool **operator**< (unsigned long long __lhs, decimal64 __rhs)
- bool **operator==** (decimal128 __lhs, decimal128 __rhs)
- bool **operator==** (decimal128 __lhs, decimal32 __rhs)
- bool **operator==** (decimal128 __lhs, decimal64 __rhs)
- bool **operator==** (decimal128 __lhs, int __rhs)
- bool **operator==** (decimal128 __lhs, long __rhs)
- bool **operator==** (decimal128 __lhs, long long __rhs)
- bool **operator==** (decimal128 __lhs, unsigned int __rhs)
- bool **operator==** (decimal128 __lhs, unsigned long __rhs)
- bool **operator==** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator==** (decimal32 __lhs, decimal128 __rhs)
- bool **operator==** (decimal32 __lhs, decimal32 __rhs)
- bool **operator==** (decimal32 __lhs, decimal64 __rhs)
- bool **operator==** (decimal32 __lhs, int __rhs)
- bool **operator==** (decimal32 __lhs, long __rhs)
- bool **operator==** (decimal32 __lhs, long long __rhs)
- bool **operator==** (decimal32 __lhs, unsigned int __rhs)

- bool **operator==** (decimal32 __lhs, unsigned long __rhs)
- bool **operator==** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator==** (decimal64 __lhs, decimal128 __rhs)
- bool **operator==** (decimal64 __lhs, decimal32 __rhs)
- bool **operator==** (decimal64 __lhs, decimal64 __rhs)
- bool **operator==** (decimal64 __lhs, int __rhs)
- bool **operator==** (decimal64 __lhs, long __rhs)
- bool **operator==** (decimal64 __lhs, long long __rhs)
- bool **operator==** (decimal64 __lhs, unsigned int __rhs)
- bool **operator==** (decimal64 __lhs, unsigned long __rhs)
- bool **operator==** (decimal64 __lhs, unsigned long long __rhs)
- bool **operator==** (int __lhs, decimal128 __rhs)
- bool **operator==** (int __lhs, decimal32 __rhs)
- bool **operator==** (int __lhs, decimal64 __rhs)
- bool **operator==** (long __lhs, decimal128 __rhs)
- bool **operator==** (long __lhs, decimal32 __rhs)
- bool **operator==** (long __lhs, decimal64 __rhs)
- bool **operator==** (long long __lhs, decimal128 __rhs)
- bool **operator==** (long long __lhs, decimal32 __rhs)
- bool **operator==** (long long __lhs, decimal64 __rhs)
- bool **operator==** (unsigned int __lhs, decimal128 __rhs)
- bool **operator==** (unsigned int __lhs, decimal32 __rhs)
- bool **operator==** (unsigned int __lhs, decimal64 __rhs)
- bool **operator==** (unsigned long __lhs, decimal128 __rhs)
- bool **operator==** (unsigned long __lhs, decimal32 __rhs)
- bool **operator==** (unsigned long __lhs, decimal64 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal128 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator==** (unsigned long long __lhs, decimal64 __rhs)
- bool **operator>** (decimal128 __lhs, decimal128 __rhs)
- bool **operator>** (decimal128 __lhs, decimal32 __rhs)
- bool **operator>** (decimal128 __lhs, decimal64 __rhs)
- bool **operator>** (decimal128 __lhs, int __rhs)
- bool **operator>** (decimal128 __lhs, long __rhs)
- bool **operator>** (decimal128 __lhs, long long __rhs)
- bool **operator>** (decimal128 __lhs, unsigned int __rhs)
- bool **operator>** (decimal128 __lhs, unsigned long __rhs)
- bool **operator>** (decimal128 __lhs, unsigned long long __rhs)
- bool **operator>** (decimal32 __lhs, decimal128 __rhs)
- bool **operator>** (decimal32 __lhs, decimal32 __rhs)
- bool **operator>** (decimal32 __lhs, decimal64 __rhs)
- bool **operator>** (decimal32 __lhs, int __rhs)
- bool **operator>** (decimal32 __lhs, long __rhs)
- bool **operator>** (decimal32 __lhs, long long __rhs)
- bool **operator>** (decimal32 __lhs, unsigned int __rhs)
- bool **operator>** (decimal32 __lhs, unsigned long __rhs)
- bool **operator>** (decimal32 __lhs, unsigned long long __rhs)
- bool **operator>** (decimal64 __lhs, decimal128 __rhs)
- bool **operator>** (decimal64 __lhs, decimal32 __rhs)
- bool **operator>** (decimal64 __lhs, decimal64 __rhs)
- bool **operator>** (decimal64 __lhs, int __rhs)

- bool **operator**> (decimal64 __lhs, long __rhs)
- bool **operator**> (decimal64 __lhs, long long __rhs)
- bool **operator**> (decimal64 __lhs, unsigned int __rhs)
- bool **operator**> (decimal64 __lhs, unsigned long __rhs)
- bool **operator**> (decimal64 __lhs, unsigned long long __rhs)
- bool **operator**> (int __lhs, decimal128 __rhs)
- bool **operator**> (int __lhs, decimal32 __rhs)
- bool **operator**> (int __lhs, decimal64 __rhs)
- bool **operator**> (long __lhs, decimal128 __rhs)
- bool **operator**> (long __lhs, decimal32 __rhs)
- bool **operator**> (long __lhs, decimal64 __rhs)
- bool **operator**> (long long __lhs, decimal128 __rhs)
- bool **operator**> (long long __lhs, decimal32 __rhs)
- bool **operator**> (long long __lhs, decimal64 __rhs)
- bool **operator**> (unsigned int __lhs, decimal128 __rhs)
- bool **operator**> (unsigned int __lhs, decimal32 __rhs)
- bool **operator**> (unsigned int __lhs, decimal64 __rhs)
- bool **operator**> (unsigned long __lhs, decimal128 __rhs)
- bool **operator**> (unsigned long __lhs, decimal32 __rhs)
- bool **operator**> (unsigned long __lhs, decimal64 __rhs)
- bool **operator**> (unsigned long long __lhs, decimal128 __rhs)
- bool **operator**> (unsigned long long __lhs, decimal32 __rhs)
- bool **operator**> (unsigned long long __lhs, decimal64 __rhs)
- bool **operator**>= (decimal128 __lhs, decimal128 __rhs)
- bool **operator**>= (decimal128 __lhs, decimal32 __rhs)
- bool **operator**>= (decimal128 __lhs, decimal64 __rhs)
- bool **operator**>= (decimal128 __lhs, int __rhs)
- bool **operator**>= (decimal128 __lhs, long __rhs)
- bool **operator**>= (decimal128 __lhs, long long __rhs)
- bool **operator**>= (decimal128 __lhs, unsigned int __rhs)
- bool **operator**>= (decimal128 __lhs, unsigned long __rhs)
- bool **operator**>= (decimal128 __lhs, unsigned long long __rhs)
- bool **operator**>= (decimal32 __lhs, decimal128 __rhs)
- bool **operator**>= (decimal32 __lhs, decimal32 __rhs)
- bool **operator**>= (decimal32 __lhs, decimal64 __rhs)
- bool **operator**>= (decimal32 __lhs, int __rhs)
- bool **operator**>= (decimal32 __lhs, long __rhs)
- bool **operator**>= (decimal32 __lhs, long long __rhs)
- bool **operator**>= (decimal32 __lhs, unsigned int __rhs)
- bool **operator**>= (decimal32 __lhs, unsigned long __rhs)
- bool **operator**>= (decimal32 __lhs, unsigned long long __rhs)
- bool **operator**>= (decimal64 __lhs, decimal128 __rhs)
- bool **operator**>= (decimal64 __lhs, decimal32 __rhs)
- bool **operator**>= (decimal64 __lhs, decimal64 __rhs)
- bool **operator**>= (decimal64 __lhs, int __rhs)
- bool **operator**>= (decimal64 __lhs, long __rhs)
- bool **operator**>= (decimal64 __lhs, long long __rhs)
- bool **operator**>= (decimal64 __lhs, unsigned int __rhs)
- bool **operator**>= (decimal64 __lhs, unsigned long __rhs)
- bool **operator**>= (decimal64 __lhs, unsigned long long __rhs)
- bool **operator**>= (int __lhs, decimal128 __rhs)

- bool **operator>=** (int __lhs, decimal32 __rhs)
- bool **operator>=** (int __lhs, decimal64 __rhs)
- bool **operator>=** (long __lhs, decimal128 __rhs)
- bool **operator>=** (long __lhs, decimal32 __rhs)
- bool **operator>=** (long __lhs, decimal64 __rhs)
- bool **operator>=** (long long __lhs, decimal128 __rhs)
- bool **operator>=** (long long __lhs, decimal32 __rhs)
- bool **operator>=** (long long __lhs, decimal64 __rhs)
- bool **operator>=** (unsigned int __lhs, decimal128 __rhs)
- bool **operator>=** (unsigned int __lhs, decimal32 __rhs)
- bool **operator>=** (unsigned int __lhs, decimal64 __rhs)
- bool **operator>=** (unsigned long __lhs, decimal128 __rhs)
- bool **operator>=** (unsigned long __lhs, decimal32 __rhs)
- bool **operator>=** (unsigned long __lhs, decimal64 __rhs)
- bool **operator>=** (unsigned long long __lhs, decimal128 __rhs)
- bool **operator>=** (unsigned long long __lhs, decimal32 __rhs)
- bool **operator>=** (unsigned long long __lhs, decimal64 __rhs)

4.15.1 Detailed Description

ISO/IEC TR 24733 Decimal floating-point arithmetic.

4.15.2 Function Documentation

decimal32_to_long_long()

```
long long std::decimal::decimal32_to_long_long (
            decimal32 __d)
```

Non-conforming extension: Conversion to integral type.

4.16 std::experimental Namespace Reference

Classes

- class any
- class bad_any_cast
- class bad_optional_access
- class basic_string_view
- struct in_place_t
- struct nullopt_t
- class optional
- class ostream_joiner
- struct owner_less< shared_ptr< _Tp > >
- struct owner_less< weak_ptr< _Tp > >
- class propagate_const

Typedefs

- template<typename _RAIter, typename _Hash, typename _Pred, typename _Val = typename iterator_traits<_RAIter>::value_type, typename _Diff = typename iterator_traits<_RAIter>::difference_type>
  using **__boyer_moore_base_t**
- template<typename _Tp>
  using **__propagate_const_elem_type**

- using **erased_type**
- using **string_view**
- using **u16string_view**
- using **u32string_view**
- using **wstring_view**

Functions

- template<typename _Fn, typename _Tuple, std::size_t... _Idx>
  constexpr decltype(auto) **__apply_impl** (_Fn &&__f, _Tuple &&__t, std::index_sequence< _Idx... >)
- template<typename _Tp, size_t _Nm, size_t... _Idx>
  constexpr array< remove_cv_t< _Tp >, _Nm > **__to_array** (_Tp(&__a)[_Nm], index_sequence< _Idx... >)
- std::default_random_engine & **_S_randint_engine** ()
- template<typename _ValueType>
  _ValueType any_cast (const any &__any)
- template<typename _Fn, typename _Tuple>
  constexpr decltype(auto) **apply** (_Fn &&__f, _Tuple &&__t)
- template<typename _Tp>
  bool **atomic_compare_exchange_strong** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp>
  bool **atomic_compare_exchange_strong_explicit** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  bool **atomic_compare_exchange_weak** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp>
  bool **atomic_compare_exchange_weak_explicit** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  void **atomic_exchange** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp>
  shared_ptr< _Tp > **atomic_exchange_explicit** (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)
- template<typename _Tp>
  bool **atomic_is_lock_free** (const shared_ptr< _Tp > *__p)
- template<typename _Tp>
  shared_ptr< _Tp > **atomic_load** (const shared_ptr< _Tp > *__p)
- template<typename _Tp>
  shared_ptr< _Tp > **atomic_load_explicit** (const shared_ptr< _Tp > *__p, memory_order __mo)
- template<typename _Tp>
  void **atomic_store** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp>
  shared_ptr< _Tp > **atomic_store_explicit** (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **const_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **dynamic_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _CharT, typename _Traits, typename _Alloc, typename _Up>
  void **erase** (basic_string< _CharT, _Traits, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Up>
  void **erase** (deque< _Tp, _Alloc > &__cont, const _Up &__value)

- template<typename _Tp, typename _Alloc, typename _Up>
  void **erase** (forward_list< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Up>
  void **erase** (list< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Up>
  void **erase** (vector< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _Predicate>
  void **erase_if** (basic_string< _CharT, _Traits, _Alloc > &__cont, _Predicate __pred)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **erase_if** (deque< _Tp, _Alloc > &__cont, _Predicate __pred)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **erase_if** (forward_list< _Tp, _Alloc > &__cont, _Predicate __pred)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **erase_if** (list< _Tp, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>
  void **erase_if** (map< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>
  void **erase_if** (multimap< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>
  void **erase_if** (multiset< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>
  void **erase_if** (set< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **erase_if** (unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **erase_if** (unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **erase_if** (unordered_multiset< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **erase_if** (unordered_set< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **erase_if** (vector< _Tp, _Alloc > &__cont, _Predicate __pred)
- template<typename _Mn, typename _Nn>
  constexpr common_type_t< _Mn, _Nn > gcd (_Mn __m, _Nn __n) noexcept
- template<typename _Del, typename _Tp>
  _Del * get_deleter (const shared_ptr< _Tp > &__p) noexcept
- template<typename _Tp>
  constexpr const _Tp & **get_underlying** (const propagate_const< _Tp > &__pt) noexcept
- template<typename _Tp>
  constexpr _Tp & **get_underlying** (propagate_const< _Tp > &__pt) noexcept
- template<typename _Mn, typename _Nn>
  constexpr common_type_t< _Mn, _Nn > lcm (_Mn __m, _Nn __n)
- template<typename _Dest = void, typename... _Types>
  constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> make_array (↩
  _Types &&... __t)
- template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _Binary↩
  Predicate = equal_to<>>
  boyer_moore_horspool_searcher< _RAIter, _Hash, _BinaryPredicate > make_boyer_moore_horspool_searcher
  (_RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())
- template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _Binary↩
  Predicate = equal_to<>>
  boyer_moore_searcher< _RAIter, _Hash, _BinaryPredicate > make_boyer_moore_searcher (_RAIter __pat_↩
  first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())

- template<typename _ForwardIterator, typename _BinaryPredicate = std::equal_to<>>
  default_searcher< _ForwardIterator, _BinaryPredicate > make_default_searcher (_ForwardIterator __pat_first, _ForwardIterator __pat_last, _BinaryPredicate __pred=_BinaryPredicate())
- template<typename _Tp>
  constexpr observer_ptr< _Tp > **make_observer** (_Tp *__p) noexcept
- template<typename _CharT, typename _Traits, typename _DelimT>
  ostream_joiner< decay_t< _DelimT >, _CharT, _Traits > make_ostream_joiner (basic_ostream< _CharT, _↩ Traits > &__os, _DelimT &&__delimiter)
- template<typename _Fn>
  auto not_fn (_Fn &&__fn) noexcept(std::is_nothrow_constructible< std::decay_t< _Fn >, _Fn && >::value)
- template<typename _CharT, typename _Traits>
  constexpr bool **operator!=** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator!=** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator!=** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator!=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **operator!=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **operator!=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr bool **operator!=** (const propagate_const< _Tp > &__pt, nullptr_t)
- template<typename _Tp>
  bool **operator!=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **operator!=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  constexpr bool **operator!=** (nullptr_t, const propagate_const< _Tp > &__pu)
- template<typename _Tp>
  bool **operator!=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp>
  constexpr bool **operator!=** (nullptr_t, observer_ptr< _Tp > __p) noexcept
- template<typename _Tp>
  bool **operator!=** (observer_ptr< _Tp > __p, nullptr_t) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator!=** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- constexpr basic_string_view< char > **operator""sv** (const char *__str, size_t __len) noexcept
- constexpr basic_string_view< char16_t > **operator""sv** (const char16_t *__str, size_t __len) noexcept
- constexpr basic_string_view< char32_t > **operator""sv** (const char32_t *__str, size_t __len) noexcept
- constexpr basic_string_view< wchar_t > **operator""sv** (const wchar_t *__str, size_t __len) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator<** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator<** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept

- template<typename _CharT, typename _Traits>
  constexpr bool **operator<** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator<** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **operator<** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **operator<** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  bool **operator<** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **operator<** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **operator<** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator<** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator<<** (basic_ostream< _CharT, _Traits > &__os, basic_string_view< _CharT, _Traits > __str)
- template<typename _Ch, typename _Tr, typename _Tp>
  std::basic_ostream< _Ch, _Tr > & **operator<<** (std::basic_ostream< _Ch, _Tr > &__os, const shared_ptr< _Tp > &__p)
- template<typename _CharT, typename _Traits>
  constexpr bool **operator<=** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator<=** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator<=** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator<=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **operator<=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **operator<=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  bool **operator<=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **operator<=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **operator<=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator<=** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _CharT, typename _Traits>
  constexpr bool **operator==** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator==** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept

- template<typename _CharT, typename _Traits>
  constexpr bool **operator==** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator==** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **operator==** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **operator==** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr bool **operator==** (const propagate_const< _Tp > &__pt, nullptr_t)
- template<typename _Tp>
  bool **operator==** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **operator==** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  constexpr bool **operator==** (nullptr_t, const propagate_const< _Tp > &__pu)
- template<typename _Tp>
  bool **operator==** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp>
  constexpr bool **operator==** (nullptr_t, observer_ptr< _Tp > __p) noexcept
- template<typename _Tp>
  constexpr bool **operator==** (observer_ptr< _Tp > __p, nullptr_t) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator==** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _CharT, typename _Traits>
  constexpr bool **operator>** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator>** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator>** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator>** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **operator>** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **operator>** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  bool **operator>** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **operator>** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **operator>** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator>** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _CharT, typename _Traits>
  constexpr bool **operator>=** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept

- template<typename _CharT, typename _Traits>
  constexpr bool **operator>=** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **operator>=** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator>=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **operator>=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **operator>=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  bool **operator>=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **operator>=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **operator>=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **operator>=** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _IntType>
  _IntType **randint** (_IntType __a, _IntType __b)
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **reinterpret_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- void **reseed** ()
- void **reseed** (default_random_engine::result_type __value)
- template<typename _PopulationIterator, typename _SampleIterator, typename _Distance>
  _SampleIterator **sample** (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n)
- template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomNumberGenerator>
  _SampleIterator sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n, _UniformRandomNumberGenerator &&__g)
- template<typename _ForwardIterator, typename _Searcher>
  _ForwardIterator **search** (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)
- template<typename _RandomAccessIterator>
  void **shuffle** (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **static_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- void swap (any &__x, any &__y) noexcept
- template<typename _Tp>
  constexpr void **swap** (observer_ptr< _Tp > &__p1, observer_ptr< _Tp > &__p2) noexcept
- template<typename _Tp>
  constexpr enable_if_t< __is_swappable< _Tp >::value, void > **swap** (propagate_const< _Tp > &__pt, propagate_const< _Tp > &__pt2) noexcept(__is_nothrow_swappable< _Tp >::value)
- template<typename _Tp>
  void **swap** (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept
- template<typename _Tp>
  void **swap** (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept
- template<typename _Tp, size_t _Nm>
  constexpr array< remove_cv_t< _Tp >, _Nm > to_array (_Tp(&__a)[_Nm]) noexcept(is_nothrow_constructible< remove_cv_t< _Tp >, _Tp & >::value)

- template<typename _ValueType>
  _ValueType any_cast (any &__any)
- template<typename _ValueType, typename enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>
  _ValueType any_cast (any &&__any)
- template<typename _ValueType, typename enable_if< is_move_constructible< _ValueType >::value &&!is_lvalue_reference< _ValueType >::value, bool >::type = false>
  _ValueType any_cast (any &&__any)

- template<typename _ValueType>
  const _ValueType * any_cast (const any *__any) noexcept
- template<typename _ValueType>
  _ValueType * any_cast (any *__any) noexcept

Variables

- template<typename _Yp, typename _Tp>
  constexpr bool **__sp_compatible_v**
- template<typename _Tp, typename _Yp>
  constexpr bool **__sp_is_constructible_v**
- constexpr in_place_t in_place
- template<typename _Tp>
  constexpr bool is_bind_expression_v
- template<typename _Tp>
  constexpr bool **is_error_code_enum_v**
- template<typename _Tp>
  constexpr bool **is_error_condition_enum_v**
- template<typename _Tp>
  constexpr int is_placeholder_v
- constexpr nullopt_t nullopt
- template<typename _R1, typename _R2>
  constexpr bool **ratio_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_greater_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_greater_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_less_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_less_v**
- template<typename _R1, typename _R2>
  constexpr bool **ratio_not_equal_v**
- template<typename _Tp>
  constexpr size_t **tuple_size_v**

- template<typename _Tp>
  constexpr bool **is_void_v**
- template<typename _Tp>
  constexpr bool **is_null_pointer_v**
- template<typename _Tp>
  constexpr bool **is_integral_v**

- template<typename _Tp>
  constexpr bool **is_floating_point_v**
- template<typename _Tp>
  constexpr bool **is_array_v**
- template<typename _Tp>
  constexpr bool **is_pointer_v**
- template<typename _Tp>
  constexpr bool **is_lvalue_reference_v**
- template<typename _Tp>
  constexpr bool **is_rvalue_reference_v**
- template<typename _Tp>
  constexpr bool **is_member_object_pointer_v**
- template<typename _Tp>
  constexpr bool **is_member_function_pointer_v**
- template<typename _Tp>
  constexpr bool **is_enum_v**
- template<typename _Tp>
  constexpr bool **is_union_v**
- template<typename _Tp>
  constexpr bool **is_class_v**
- template<typename _Tp>
  constexpr bool **is_function_v**
- template<typename _Tp>
  constexpr bool **is_reference_v**
- template<typename _Tp>
  constexpr bool **is_arithmetic_v**
- template<typename _Tp>
  constexpr bool **is_fundamental_v**
- template<typename _Tp>
  constexpr bool **is_object_v**
- template<typename _Tp>
  constexpr bool **is_scalar_v**
- template<typename _Tp>
  constexpr bool **is_compound_v**
- template<typename _Tp>
  constexpr bool **is_member_pointer_v**
- template<typename _Tp>
  constexpr bool **is_const_v**
- template<typename _Tp>
  constexpr bool **is_volatile_v**
- template<typename _Tp>
  constexpr bool **is_trivial_v**
- template<typename _Tp>
  constexpr bool **is_trivially_copyable_v**
- template<typename _Tp>
  constexpr bool **is_standard_layout_v**
- template<typename _Tp>
  constexpr bool **is_pod_v**
- template<typename _Tp>
  constexpr bool **is_literal_type_v**
- template<typename _Tp>
  constexpr bool **is_empty_v**

- template<typename _Tp>
  constexpr bool **is_polymorphic_v**
- template<typename _Tp>
  constexpr bool **is_abstract_v**
- template<typename _Tp>
  constexpr bool **is_final_v**
- template<typename _Tp>
  constexpr bool **is_signed_v**
- template<typename _Tp>
  constexpr bool **is_unsigned_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **is_constructible_v**
- template<typename _Tp>
  constexpr bool **is_default_constructible_v**
- template<typename _Tp>
  constexpr bool **is_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **is_move_constructible_v**
- template<typename _Tp, typename _Up>
  constexpr bool **is_assignable_v**
- template<typename _Tp>
  constexpr bool **is_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **is_move_assignable_v**
- template<typename _Tp>
  constexpr bool **is_destructible_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **is_trivially_constructible_v**
- template<typename _Tp>
  constexpr bool **is_trivially_default_constructible_v**
- template<typename _Tp>
  constexpr bool **is_trivially_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **is_trivially_move_constructible_v**
- template<typename _Tp, typename _Up>
  constexpr bool **is_trivially_assignable_v**
- template<typename _Tp>
  constexpr bool **is_trivially_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **is_trivially_move_assignable_v**
- template<typename _Tp>
  constexpr bool **is_trivially_destructible_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **is_nothrow_constructible_v**
- template<typename _Tp>
  constexpr bool **is_nothrow_default_constructible_v**
- template<typename _Tp>
  constexpr bool **is_nothrow_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **is_nothrow_move_constructible_v**
- template<typename _Tp, typename _Up>
  constexpr bool **is_nothrow_assignable_v**

- template<typename _Tp>
  constexpr bool **is_nothrow_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **is_nothrow_move_assignable_v**
- template<typename _Tp>
  constexpr bool **is_nothrow_destructible_v**
- template<typename _Tp>
  constexpr bool **has_virtual_destructor_v**
- template<typename _Tp>
  constexpr size_t **alignment_of_v**
- template<typename _Tp>
  constexpr size_t **rank_v**
- template<typename _Tp, unsigned _Idx = 0>
  constexpr size_t **extent_v**
- template<typename _Tp, typename _Up>
  constexpr bool **is_same_v**
- template<typename _Tp>
  constexpr bool **is_same_v**< **_Tp, _Tp** >
- template<typename _Base, typename _Derived>
  constexpr bool **is_base_of_v**
- template<typename _From, typename _To>
  constexpr bool **is_convertible_v**

- template<typename... _Bn>
  constexpr bool **conjunction_v**
- template<typename... _Bn>
  constexpr bool **disjunction_v**
- template<typename _Pp>
  constexpr bool **negation_v**

- template<typename...>
  using void_t
- template<typename _Default, template< typename... > class _Op, typename... _Args>
  using detected_or
- template<typename _Default, template< typename... > class _Op, typename... _Args>
  using detected_or_t
- template<template< typename... > class _Op, typename... _Args>
  using detected_t
- template<template< typename... > class _Op, typename... _Args>
  using is_detected
- template<typename _Expected, template< typename... > class _Op, typename... _Args>
  using is_detected_exact
- template<typename _To, template< typename... > class _Op, typename... _Args>
  using is_detected_convertible
- template<template< typename... > class _Op, typename... _Args>
  constexpr bool is_detected_v
- template<typename _Expected, template< typename... > class _Op, typename... _Args>
  constexpr bool is_detected_exact_v
- template<typename _To, template< typename... > class _Op, typename... _Args>
  constexpr bool is_detected_convertible_v

4.16.1 Detailed Description

Namespace for features defined in ISO Technical Specifications.

4.16.2 Function Documentation

gcd()

```
template<typename _Mn, typename _Nn>
common_type_t< _Mn, _Nn > std::experimental::fundamentals_v2::gcd (
            _Mn __m,
            _Nn __n)  [constexpr], [noexcept]
```

Greatest common divisor.

get_deleter()

```
template<typename _Del, typename _Tp>
_Del * std::experimental::fundamentals_v2::get_deleter (
            const shared_ptr< _Tp > & __p)  [inline], [noexcept]
```

C++14 20.8.2.2.10.

lcm()

```
template<typename _Mn, typename _Nn>
common_type_t< _Mn, _Nn > std::experimental::fundamentals_v2::lcm (
            _Mn __m,
            _Nn __n)  [constexpr]
```

Least common multiple.

make_boyer_moore_horspool_searcher()

```
template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::↩
value_type>, typename _BinaryPredicate = equal_to<>>
boyer_moore_horspool_searcher< _RAIter, _Hash, _BinaryPredicate > std::experimental::fundamentals↩
_v1::make_boyer_moore_horspool_searcher (
            _RAIter __pat_first,
            _RAIter __pat_last,
            _Hash __hf = _Hash(),
            _BinaryPredicate __pred = _BinaryPredicate())  [inline]
```

Generator function for boyer_moore_horspool_searcher.
References std::move().

make_boyer_moore_searcher()

```
template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::↩
value_type>, typename _BinaryPredicate = equal_to<>>
boyer_moore_searcher< _RAIter, _Hash, _BinaryPredicate > std::experimental::fundamentals_v1::↩
make_boyer_moore_searcher (
            _RAIter __pat_first,
            _RAIter __pat_last,
            _Hash __hf = _Hash(),
            _BinaryPredicate __pred = _BinaryPredicate())  [inline]
```

Generator function for boyer_moore_searcher.
References std::move().

make_default_searcher()

```
template<typename _ForwardIterator, typename _BinaryPredicate = std::equal_to<>>
default_searcher< _ForwardIterator, _BinaryPredicate > std::experimental::fundamentals_v1::make↩
_default_searcher (
            _ForwardIterator __pat_first,
            _ForwardIterator __pat_last,
            _BinaryPredicate __pred = _BinaryPredicate())  [inline]
```

Generator function for default_searcher.

make_ostream_joiner()

```
template<typename _CharT, typename _Traits, typename _DelimT>
ostream_joiner< decay_t< _DelimT >, _CharT, _Traits > std::experimental::fundamentals_v2::make↩
_ostream_joiner (
            basic_ostream< _CharT, _Traits > & __os,
            _DelimT && __delimiter)  [inline]
```

Object generator for ostream_joiner.
References std::forward().

not_fn()

```
template<typename _Fn>
auto std::experimental::fundamentals_v2::not_fn (
            _Fn && __fn)  [inline], [noexcept]
```

[func.not_fn] Function template not_fn
References std::forward().

sample()

```
template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _↩
UniformRandomNumberGenerator>
_SampleIterator std::experimental::fundamentals_v2::sample (
            _PopulationIterator __first,
            _PopulationIterator __last,
            _SampleIterator __out,
            _Distance __n,
            _UniformRandomNumberGenerator && __g)
```

Take a random sample from a population.
References std::forward().

4.16.3 Variable Documentation

is_bind_expression_v

```
template<typename _Tp>
bool std::experimental::fundamentals_v1::is_bind_expression_v  [constexpr]
```

Variable template for std::is_bind_expression.

is_placeholder_v

```
template<typename _Tp>
int std::experimental::fundamentals_v1::is_placeholder_v  [constexpr]
```

Variable template for std::is_placeholder.

4.17 std::filesystem Namespace Reference

Classes

- class directory_entry
- class directory_iterator
- class file_status
- class filesystem_error
- class path
- class recursive_directory_iterator
- struct space_info

Typedefs

- using file_time_type

Enumerations

- enum class copy_options : unsigned short {
  none , **skip_existing** , **overwrite_existing** , **update_existing** ,
  recursive , **copy_symlinks** , **skip_symlinks** , **directories_only** ,
  create_symlinks , **create_hard_links** }
- enum class directory_options : unsigned char { **none** , **follow_directory_symlink** , **skip_permission_denied** }
- enum class file_type : signed char {
  none , **not_found** , **regular** , **directory** ,
  symlink , **block** , **character** , **fifo** ,
  socket , **unknown** }
- enum class perm_options : unsigned { **replace** , **add** , **remove** , **nofollow** }
- enum class perms : unsigned {
  none , **owner_read** , **owner_write** , **owner_exec** ,
  owner_all , **group_read** , **group_write** , **group_exec** ,
  group_all , **others_read** , **others_write** , **others_exec** ,
  others_all , **all** , **set_uid** , **set_gid** ,
  sticky_bit , **mask** , **unknown** }

Functions

- path **absolute** (const path &__p)
- path **absolute** (const path &__p, error_code &__ec)
- path **canonical** (const path &__p)
- path **canonical** (const path &__p, error_code &__ec)
- void **copy** (const path &__from, const path &__to)
- void **copy** (const path &__from, const path &__to, copy_options __options)
- void **copy** (const path &__from, const path &__to, copy_options __options, error_code &)
- void **copy** (const path &__from, const path &__to, error_code &__ec)
- bool **copy_file** (const path &__from, const path &__to)
- bool **copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **copy_file** (const path &__from, const path &__to, copy_options __option, error_code &)
- bool **copy_file** (const path &__from, const path &__to, error_code &__ec)
- void **copy_symlink** (const path &__existing_symlink, const path &__new_symlink)
- void **copy_symlink** (const path &__existing_symlink, const path &__new_symlink, error_code &__ec) noexcept
- bool **create_directories** (const path &__p)
- bool **create_directories** (const path &__p, error_code &__ec)
- bool **create_directory** (const path &__p)

- bool **create_directory** (const path &__p, const path &__attributes)
- bool **create_directory** (const path &__p, const path &__attributes, error_code &__ec) noexcept
- bool **create_directory** (const path &__p, error_code &__ec) noexcept
- void **create_directory_symlink** (const path &__to, const path &__new_symlink)
- void **create_directory_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- void **create_hard_link** (const path &__to, const path &__new_hard_link)
- void **create_hard_link** (const path &__to, const path &__new_hard_link, error_code &__ec) noexcept
- void **create_symlink** (const path &__to, const path &__new_symlink)
- void **create_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- path **current_path** ()
- void **current_path** (const path &__p)
- void **current_path** (const path &__p, error_code &__ec) noexcept
- path **current_path** (error_code &__ec)
- bool **equivalent** (const path &__p1, const path &__p2)
- bool **equivalent** (const path &__p1, const path &__p2, error_code &__ec) noexcept
- bool **exists** (const path &__p)
- bool **exists** (const path &__p, error_code &__ec) noexcept
- bool **exists** (file_status) noexcept
- uintmax_t **file_size** (const path &)
- uintmax_t **file_size** (const path &, error_code &) noexcept
- uintmax_t **hard_link_count** (const path &)
- uintmax_t **hard_link_count** (const path &, error_code &) noexcept
- size_t **hash_value** (const path &__p) noexcept
- bool **is_block_file** (const path &__p)
- bool **is_block_file** (const path &__p, error_code &__ec) noexcept
- bool **is_block_file** (file_status __s) noexcept
- bool **is_character_file** (const path &__p)
- bool **is_character_file** (const path &__p, error_code &__ec) noexcept
- bool **is_character_file** (file_status __s) noexcept
- bool **is_directory** (const path &__p)
- bool **is_directory** (const path &__p, error_code &__ec) noexcept
- bool **is_directory** (file_status __s) noexcept
- bool **is_empty** (const path &__p)
- bool **is_empty** (const path &__p, error_code &__ec)
- bool **is_fifo** (const path &__p)
- bool **is_fifo** (const path &__p, error_code &__ec) noexcept
- bool **is_fifo** (file_status __s) noexcept
- bool **is_other** (const path &__p)
- bool **is_other** (const path &__p, error_code &__ec) noexcept
- bool **is_other** (file_status) noexcept
- bool **is_regular_file** (const path &__p)
- bool **is_regular_file** (const path &__p, error_code &__ec) noexcept
- bool **is_regular_file** (file_status) noexcept
- bool **is_socket** (const path &__p)
- bool **is_socket** (const path &__p, error_code &__ec) noexcept
- bool **is_socket** (file_status __s) noexcept
- bool **is_symlink** (const path &__p)
- bool **is_symlink** (const path &__p, error_code &__ec) noexcept
- bool **is_symlink** (file_status) noexcept
- file_time_type **last_write_time** (const path &)
- file_time_type **last_write_time** (const path &, error_code &) noexcept

- void **last_write_time** (const path &__p, file_time_type __new_time)
- void **last_write_time** (const path &__p, file_time_type __new_time, error_code &__ec) noexcept
- copy_options & **operator&=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **operator^** (copy_options __x, copy_options __y) noexcept
- copy_options & **operator^=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **operator|** (copy_options __x, copy_options __y) noexcept
- copy_options & **operator|=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **operator~** (copy_options __x) noexcept
- void **permissions** (const path &, perms, perm_options, error_code &) noexcept
- void **permissions** (const path &__p, perms __prms, error_code &__ec) noexcept
- void **permissions** (const path &__p, perms __prms, perm_options __opts=perm_options::replace)
- path **proximate** (const path &__p, const path &__base, error_code &__ec)
- path **proximate** (const path &__p, const path &__base=current_path())
- path **proximate** (const path &__p, error_code &__ec)
- path **read_symlink** (const path &__p)
- path **read_symlink** (const path &__p, error_code &__ec)
- path **relative** (const path &__p, const path &__base, error_code &__ec)
- path **relative** (const path &__p, const path &__base=current_path())
- path **relative** (const path &__p, error_code &__ec)
- bool **remove** (const path &, error_code &) noexcept
- bool **remove** (const path &__p)
- uintmax_t **remove_all** (const path &)
- uintmax_t **remove_all** (const path &, error_code &)
- void **rename** (const path &__from, const path &__to)
- void **rename** (const path &__from, const path &__to, error_code &__ec) noexcept
- void **resize_file** (const path &__p, uintmax_t __size)
- void **resize_file** (const path &__p, uintmax_t __size, error_code &__ec) noexcept
- space_info **space** (const path &__p)
- space_info **space** (const path &__p, error_code &__ec) noexcept
- file_status **status** (const path &)
- file_status **status** (const path &, error_code &) noexcept
- bool **status_known** (file_status) noexcept
- void **swap** (path &__lhs, path &__rhs) noexcept
- file_status **symlink_status** (const path &)
- file_status **symlink_status** (const path &, error_code &) noexcept
- path **temp_directory_path** ()
- path **temp_directory_path** (error_code &__ec)
- template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename _CharT = __detail::__value_type_↩ is_char_or_char8_t<_InputIterator>>
  path u8path (_InputIterator __first, _InputIterator __last)
- template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _CharT = __detail::__value_type_is_char_or↩ _char8_t<_Source>>
  path u8path (const _Source &__source)
- path **weakly_canonical** (const path &__p)
- path **weakly_canonical** (const path &__p, error_code &__ec)

- constexpr perms **operator|** (perms __x, perms __y) noexcept
- constexpr perms **operator^** (perms __x, perms __y) noexcept
- constexpr perms **operator~** (perms __x) noexcept
- perms & **operator&=** (perms &__x, perms __y) noexcept

- perms & **operator|=** (perms &__x, perms __y) noexcept
- perms & **operator^=** (perms &__x, perms __y) noexcept

- constexpr perm_options **operator|** (perm_options __x, perm_options __y) noexcept
- constexpr perm_options **operator^** (perm_options __x, perm_options __y) noexcept
- constexpr perm_options **operator~** (perm_options __x) noexcept
- perm_options & **operator&=** (perm_options &__x, perm_options __y) noexcept
- perm_options & **operator|=** (perm_options &__x, perm_options __y) noexcept
- perm_options & **operator^=** (perm_options &__x, perm_options __y) noexcept

- constexpr directory_options **operator|** (directory_options __x, directory_options __y) noexcept
- constexpr directory_options **operator^** (directory_options __x, directory_options __y) noexcept
- constexpr directory_options **operator~** (directory_options __x) noexcept
- directory_options & **operator&=** (directory_options &__x, directory_options __y) noexcept
- directory_options & **operator|=** (directory_options &__x, directory_options __y) noexcept
- directory_options & **operator^=** (directory_options &__x, directory_options __y) noexcept

4.17.1 Detailed Description

ISO C++ 2017 namespace for File System library.

4.18 std::literals Namespace Reference

Namespaces

- namespace chrono_literals

Functions

- constexpr chrono::day operator""d (unsigned long long __d) noexcept
- template<char... _Digits>
  constexpr chrono::hours operator""h ()
- constexpr chrono::duration< long double, ratio< 3600, 1 > > operator""h (long double __hours)
- template<char... _Digits>
  constexpr chrono::minutes operator""min ()
- constexpr chrono::duration< long double, ratio< 60, 1 > > operator""min (long double __mins)
- template<char... _Digits>
  constexpr chrono::milliseconds operator""ms ()
- constexpr chrono::duration< long double, milli > operator""ms (long double __msecs)
- template<char... _Digits>
  constexpr chrono::nanoseconds operator""ns ()
- constexpr chrono::duration< long double, nano > operator""ns (long double __nsecs)
- template<char... _Digits>
  constexpr chrono::seconds operator""s ()
- constexpr chrono::duration< long double > operator""s (long double __secs)
- constexpr basic_string_view< char > **operator""sv** (const char *__str, size_t __len) noexcept
- constexpr basic_string_view< char16_t > **operator""sv** (const char16_t *__str, size_t __len) noexcept
- constexpr basic_string_view< char32_t > **operator""sv** (const char32_t *__str, size_t __len) noexcept
- constexpr basic_string_view< wchar_t > **operator""sv** (const wchar_t *__str, size_t __len) noexcept
- template<char... _Digits>
  constexpr chrono::microseconds operator""us ()
- constexpr chrono::duration< long double, micro > operator""us (long double __usecs)
- constexpr chrono::year operator""y (unsigned long long __y) noexcept

4.18.1 Detailed Description

ISO C++ inline namespace for literal suffixes.

4.19 std::literals::chrono_literals Namespace Reference

Functions

- constexpr chrono::day operator""d (unsigned long long __d) noexcept
- template<char... _Digits>
  constexpr chrono::hours operator""h ()
- constexpr chrono::duration< long double, ratio< 3600, 1 > > operator""h (long double __hours)
- template<char... _Digits>
  constexpr chrono::minutes operator""min ()
- constexpr chrono::duration< long double, ratio< 60, 1 > > operator""min (long double __mins)
- template<char... _Digits>
  constexpr chrono::milliseconds operator""ms ()
- constexpr chrono::duration< long double, milli > operator""ms (long double __msecs)
- template<char... _Digits>
  constexpr chrono::nanoseconds operator""ns ()
- constexpr chrono::duration< long double, nano > operator""ns (long double __nsecs)
- template<char... _Digits>
  constexpr chrono::seconds operator""s ()
- constexpr chrono::duration< long double > operator""s (long double __secs)
- template<char... _Digits>
  constexpr chrono::microseconds operator""us ()
- constexpr chrono::duration< long double, micro > operator""us (long double __usecs)
- constexpr chrono::year operator""y (unsigned long long __y) noexcept

4.19.1 Detailed Description

ISO C++ 2014 namespace for suffixes for duration literals.
These suffixes can be used to create `chrono::duration` values with tick periods of hours, minutes, seconds, milliseconds, microseconds or nanoseconds. For example, `std::chrono::seconds(5)` can be written as `5s` after making the suffix visible in the current scope. The suffixes can be made visible by a using-directive or using-declaration such as:

- `using namespace std::chrono_literals;`
- `using namespace std::literals;`
- `using namespace std::chrono;`
- `using namespace std;`
- `using std::chrono_literals::operator""s;`

The result of these suffixes on an integer literal is one of the standard typedefs such as `std::chrono::hours`. The result on a floating-point literal is a duration type with the specified tick period and an unspecified floating-point representation, for example `1.5e2ms` might be equivalent to `chrono::duration<long double, chrono::milli>(1.5e2)`.

Since

C++14

4.20 std::placeholders Namespace Reference

Variables

- const _Placeholder< 1 > **_1**
- const _Placeholder< 10 > **_10**
- const _Placeholder< 11 > **_11**
- const _Placeholder< 12 > **_12**
- const _Placeholder< 13 > **_13**
- const _Placeholder< 14 > **_14**
- const _Placeholder< 15 > **_15**
- const _Placeholder< 16 > **_16**
- const _Placeholder< 17 > **_17**
- const _Placeholder< 18 > **_18**
- const _Placeholder< 19 > **_19**
- const _Placeholder< 2 > **_2**
- const _Placeholder< 20 > **_20**
- const _Placeholder< 21 > **_21**
- const _Placeholder< 22 > **_22**
- const _Placeholder< 23 > **_23**
- const _Placeholder< 24 > **_24**
- const _Placeholder< 25 > **_25**
- const _Placeholder< 26 > **_26**
- const _Placeholder< 27 > **_27**
- const _Placeholder< 28 > **_28**
- const _Placeholder< 29 > **_29**
- const _Placeholder< 3 > **_3**
- const _Placeholder< 4 > **_4**
- const _Placeholder< 5 > **_5**
- const _Placeholder< 6 > **_6**
- const _Placeholder< 7 > **_7**
- const _Placeholder< 8 > **_8**
- const _Placeholder< 9 > **_9**

4.20.1 Detailed Description

ISO C++ 2011 namespace for std::bind placeholders.

Since

C++11

4.21 std::regex_constants Namespace Reference

5.1 Regular Expression Syntax Options

- enum syntax_option_type : unsigned int { **_S_icase** , **_S_nosubs** , **_S_optimize** , **_S_collate** , **_S_ECMAScript** , **_S_basic** , **_S_extended** , **_S_awk** , **_S_grep** , **_S_egrep** , **_S_polynomial** , **_S_multiline** }
- constexpr syntax_option_type icase
- constexpr syntax_option_type nosubs
- constexpr syntax_option_type optimize
- constexpr syntax_option_type collate

- constexpr syntax_option_type ECMAScript
- constexpr syntax_option_type basic
- constexpr syntax_option_type extended
- constexpr syntax_option_type awk
- constexpr syntax_option_type grep
- constexpr syntax_option_type egrep
- constexpr syntax_option_type multiline
- constexpr syntax_option_type __multiline
- constexpr syntax_option_type __polynomial
- constexpr syntax_option_type operator& (syntax_option_type __a, syntax_option_type __b) noexcept
- constexpr syntax_option_type operator| (syntax_option_type __a, syntax_option_type __b) noexcept
- constexpr syntax_option_type operator^ (syntax_option_type __a, syntax_option_type __b) noexcept
- constexpr syntax_option_type operator~ (syntax_option_type __a) noexcept
- constexpr syntax_option_type & operator&= (syntax_option_type &__a, syntax_option_type __b) noexcept
- constexpr syntax_option_type & operator|= (syntax_option_type &__a, syntax_option_type __b) noexcept
- constexpr syntax_option_type & operator^= (syntax_option_type &__a, syntax_option_type __b) noexcept

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- enum match_flag_type : unsigned int {
  _S_default , **_S_not_bol** , **_S_not_eol** , **_S_not_bow** ,
  _S_not_eow , **_S_any** , **_S_not_null** , **_S_continuous** ,
  _S_prev_avail , **_S_sed** , **_S_no_copy** , **_S_first_only** ,
  _S_match_flag_last }
- constexpr match_flag_type match_default
- constexpr match_flag_type match_not_bol
- constexpr match_flag_type match_not_eol
- constexpr match_flag_type match_not_bow
- constexpr match_flag_type match_not_eow
- constexpr match_flag_type match_any
- constexpr match_flag_type match_not_null
- constexpr match_flag_type match_continuous
- constexpr match_flag_type match_prev_avail
- constexpr match_flag_type format_default
- constexpr match_flag_type format_sed
- constexpr match_flag_type format_no_copy
- constexpr match_flag_type format_first_only
- constexpr match_flag_type operator& (match_flag_type __a, match_flag_type __b) noexcept
- constexpr match_flag_type operator| (match_flag_type __a, match_flag_type __b) noexcept
- constexpr match_flag_type operator^ (match_flag_type __a, match_flag_type __b) noexcept
- constexpr match_flag_type operator~ (match_flag_type __a) noexcept
- constexpr match_flag_type & operator&= (match_flag_type &__a, match_flag_type __b) noexcept
- constexpr match_flag_type & operator|= (match_flag_type &__a, match_flag_type __b) noexcept
- constexpr match_flag_type & operator^= (match_flag_type &__a, match_flag_type __b) noexcept

5.3 Error Types

- enum error_type {
  _S_error_collate , **_S_error_ctype** , **_S_error_escape** , **_S_error_backref** ,
  _S_error_brack , **_S_error_paren** , **_S_error_brace** , **_S_error_badbrace** ,
  _S_error_range , **_S_error_space** , **_S_error_badrepeat** , **_S_error_complexity** ,
  _S_error_stack , **_S_null** , **_S_grammar** }
- constexpr error_type error_collate (_S_error_collate)
- constexpr error_type error_ctype (_S_error_ctype)
- constexpr error_type error_escape (_S_error_escape)
- constexpr error_type error_backref (_S_error_backref)
- constexpr error_type error_brack (_S_error_brack)
- constexpr error_type error_paren (_S_error_paren)
- constexpr error_type error_brace (_S_error_brace)
- constexpr error_type error_badbrace (_S_error_badbrace)
- constexpr error_type error_range (_S_error_range)
- constexpr error_type error_space (_S_error_space)
- constexpr error_type error_badrepeat (_S_error_badrepeat)
- constexpr error_type error_complexity (_S_error_complexity)
- constexpr error_type error_stack (_S_error_stack)

4.21.1 Detailed Description

ISO C++ 2011 namespace for options and flags used with std::regex.

4.21.2 Enumeration Type Documentation

error_type

```
enum std::regex_constants::error_type
```

The expression contained an invalid collating element name.

match_flag_type

```
enum std::regex_constants::match_flag_type :  unsigned int
```

This is a bitmask type indicating regex matching rules.
The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

syntax_option_type

```
enum std::regex_constants::syntax_option_type :  unsigned int
```

This is a bitmask type indicating how to interpret the regex.
The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

4.21.3 Function Documentation

error_backref()

```
error_type std::regex_constants::error_backref (
            _S_error_backref )  [inline], [constexpr]
```

The expression contained an invalid back reference.

error_badbrace()

```
error_type std::regex_constants::error_badbrace (
            _S_error_badbrace )  [inline], [constexpr]
```

The expression contained an invalid range in a {} expression.

error_badrepeat()

```
error_type std::regex_constants::error_badrepeat (
            _S_error_badrepeat )  [inline], [constexpr]
```

One of *∗?+{* was not preceded by a valid regular expression.

error_brace()

```
error_type std::regex_constants::error_brace (
            _S_error_brace )  [inline], [constexpr]
```

The expression contained mismatched { and }

error_brack()

```
error_type std::regex_constants::error_brack (
            _S_error_brack )  [inline], [constexpr]
```

The expression contained mismatched [and].

error_collate()

```
error_type std::regex_constants::error_collate (
            _S_error_collate )  [inline], [constexpr]
```

The expression contained an invalid collating element name.

error_complexity()

```
error_type std::regex_constants::error_complexity (
            _S_error_complexity )  [inline], [constexpr]
```

The complexity of an attempted match against a regular expression exceeded a pre-set level.

error_ctype()

```
error_type std::regex_constants::error_ctype (
            _S_error_ctype )  [inline], [constexpr]
```

The expression contained an invalid character class name.

error_escape()

```
error_type std::regex_constants::error_escape (
            _S_error_escape )  [inline], [constexpr]
```

The expression contained an invalid escaped character, or a trailing escape.

error_paren()

```
error_type std::regex_constants::error_paren (
            _S_error_paren )  [inline], [constexpr]
```

The expression contained mismatched (and).

error_range()

```
error_type std::regex_constants::error_range (
            _S_error_range )  [inline], [constexpr]
```

The expression contained an invalid character range, such as [b-a] in most encodings.

error_space()

```
error_type std::regex_constants::error_space (
            _S_error_space )  [inline], [constexpr]
```

There was insufficient memory to convert the expression into a finite state machine.

error_stack()

```
error_type std::regex_constants::error_stack (
            _S_error_stack )  [inline], [constexpr]
```

There was insufficient memory to determine whether the regular expression could match the specified character sequence.

operator&() [1/2]

```
match_flag_type std::regex_constants::operator& (
            match_flag_type __a,
            match_flag_type __b)  [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator&() [2/2]

```
syntax_option_type std::regex_constants::operator& (
            syntax_option_type __a,
            syntax_option_type __b)  [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.
The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements `ECMAScript`, `basic`, `extended`, `awk`, `grep`, `egrep` set.

operator&=() [1/2]

```
match_flag_type & std::regex_constants::operator&= (
            match_flag_type & __a,
            match_flag_type __b)  [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The `match_flag_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator&=() [2/2]

```
syntax_option_type & std::regex_constants::operator&= (
            syntax_option_type & __a,
            syntax_option_type __b)  [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.
The `syntax_option_type` is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

A valid value of type syntax_option_type shall have exactly one of the elements ECMAScript, basic, extended, awk, grep, egrep set.

operator^() [1/2]

```
match_flag_type std::regex_constants::operator^ (
            match_flag_type __a,
            match_flag_type __b)  [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The match_flag_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator^() [2/2]

```
syntax_option_type std::regex_constants::operator^ (
            syntax_option_type __a,
            syntax_option_type __b)  [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.
The syntax_option_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements ECMAScript, basic, extended, awk, grep, egrep set.

operator^=() [1/2]

```
match_flag_type & std::regex_constants::operator^= (
            match_flag_type & __a,
            match_flag_type __b)  [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The match_flag_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator^=() [2/2]

```
syntax_option_type & std::regex_constants::operator^= (
            syntax_option_type & __a,
            syntax_option_type __b)  [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.
The syntax_option_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements ECMAScript, basic, extended, awk, grep, egrep set.

operator"|() [1/2]

```
match_flag_type std::regex_constants::operator| (
            match_flag_type __a,
            match_flag_type __b)  [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The match_flag_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator"|() [2/2]

```
syntax_option_type std::regex_constants::operator| (
```

```
            syntax_option_type __a,
            syntax_option_type __b)  [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.
The syntax_option_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements ECMAScript, basic, extended, awk, grep, egrep set.

operator"|=() [1/2]

```
match_flag_type & std::regex_constants::operator|= (
            match_flag_type & __a,
            match_flag_type __b)  [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The match_flag_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator"|=() [2/2]

```
syntax_option_type & std::regex_constants::operator|= (
            syntax_option_type & __a,
            syntax_option_type __b)  [inline], [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.
The syntax_option_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements ECMAScript, basic, extended, awk, grep, egrep set.

operator~() [1/2]

```
match_flag_type std::regex_constants::operator~ (
            match_flag_type __a)  [constexpr], [noexcept]
```

This is a bitmask type indicating regex matching rules.
The match_flag_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.

operator~() [2/2]

```
syntax_option_type std::regex_constants::operator~ (
            syntax_option_type __a)  [constexpr], [noexcept]
```

This is a bitmask type indicating how to interpret the regex.
The syntax_option_type is implementation defined but it is valid to perform bitwise operations on these values and expect the right thing to happen.
A valid value of type syntax_option_type shall have exactly one of the elements ECMAScript, basic, extended, awk, grep, egrep set.

4.21.4 Variable Documentation

__multiline

```
syntax_option_type std::regex_constants::__multiline  [inline], [constexpr]
```

Extension: Equivalent to regex_constants::multiline for C++11 and C++14.

__polynomial

```
syntax_option_type std::regex_constants::__polynomial  [inline], [constexpr]
```

Extension: Ensure both space complexity of compiled regex and time complexity execution are not exponential. If specified in a regex with back-references, the exception regex_constants::error_complexity will be thrown.

awk

syntax_option_type std::regex_constants::awk [inline], [constexpr]
Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility awk in IEEE Std 1003.1-2001. This option is identical to syntax_option_type extended, except that C-style escape sequences are supported. These sequences are: \\, \a, \b, \f, \n, \r, \t , \v, \&apos„ &apos„ and \ddd (where ddd is one, two, or three octal digits).

basic

syntax_option_type std::regex_constants::basic [inline], [constexpr]
Specifies that the grammar recognized by the regular expression engine is that used by POSIX basic regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

collate

syntax_option_type std::regex_constants::collate [inline], [constexpr]
Specifies that character ranges of the form [a-b] should be locale sensitive.

ECMAScript

syntax_option_type std::regex_constants::ECMAScript [inline], [constexpr]
Specifies that the grammar recognized by the regular expression engine is that used by ECMAScript in ECMA-262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], as modified in section [28.13]. This grammar is similar to that defined in the PERL scripting language but extended with elements found in the POSIX regular expression grammar.

egrep

syntax_option_type std::regex_constants::egrep [inline], [constexpr]
Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility grep when given the -E option in IEEE Std 1003.1-2001. This option is identical to syntax_option_type extended, except that newlines are treated as whitespace.

extended

syntax_option_type std::regex_constants::extended [inline], [constexpr]
Specifies that the grammar recognized by the regular expression engine is that used by POSIX extended regular expressions in IEEE Std 1003.1-2001, Portable Operating System Interface (POSIX), Base Definitions and Headers, Section 9, Regular Expressions.

format_default

match_flag_type std::regex_constants::format_default [inline], [constexpr]
When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the ECMAScript replace function in ECMA- 262 [Ecma International, ECMAScript Language Specification, Standard Ecma-262, third edition, 1999], part 15.5.4.11 String.prototype.replace. In addition, during search and replace operations all non-overlapping occurrences of the regular expression are located and replaced, and sections of the input that did not match the expression are copied unchanged to the output string.
Format strings (from ECMA-262 [15.5.4.11]):

- $$ The dollar-sign itself ($)
- $& The matched substring.
- $` The portion of the string that precedes the matched substring. This would be match_results::prefix().
- $' The portion of the string that follows the matched substring. This would be match_results::suffix().
- $n The nth capture, where n is a positive integer less than 100. If `n <= match_results::size()` and the nth capture is undefined, use the empty string instead. If `n > match_results::size()`, the result is implementation-defined.

Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::format(), std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::format(), std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::format(), and std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::format().

format_first_only

`match_flag_type std::regex_constants::format_first_only [inline], [constexpr]`
When specified during a search and replace operation, only the first occurrence of the regular expression shall be replaced.

format_no_copy

`match_flag_type std::regex_constants::format_no_copy [inline], [constexpr]`
During a search and replace operation, sections of the character container sequence being searched that do not match the regular expression shall not be copied to the output string.

format_sed

`match_flag_type std::regex_constants::format_sed [inline], [constexpr]`
When a regular expression match is to be replaced by a new string, the new string is constructed using the rules used by the POSIX sed utility in IEEE Std 1003.1- 2001 [IEEE, Information Technology – Portable Operating System Interface (POSIX), IEEE Standard 1003.1-2001].

grep

`syntax_option_type std::regex_constants::grep [inline], [constexpr]`
Specifies that the grammar recognized by the regular expression engine is that used by POSIX utility grep in IEEE Std 1003.1-2001. This option is identical to syntax_option_type basic, except that newlines are treated as whitespace.

icase

`syntax_option_type std::regex_constants::icase [inline], [constexpr]`
Specifies that the matching of regular expressions against a character sequence shall be performed without regard to case.

match_any

`match_flag_type std::regex_constants::match_any [inline], [constexpr]`
If more than one match is possible then any match is an acceptable result.

match_continuous

match_flag_type std::regex_constants::match_continuous [inline], [constexpr]
The expression only matches a sub-sequence that begins at first .
Referenced by std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++().

match_default

match_flag_type std::regex_constants::match_default [inline], [constexpr]
The default matching rules.
Referenced by std::regex_iterator< const char * >::regex_iterator(), std::regex_token_iterator< const char * >::regex_token_iterator(), std::regex_token_iterator< const char * >::regex_token_iterator(), std::regex_token_iterator< const char * >::regex_token_iterator(), std::regex_token_iterator< const char * >::regex_token_iterator(), std::regex_match(), std::regex_match(), std::regex_match(), std::regex_match(), std::regex_match(), std::regex_match(), std::regex_match(), std::regex_replace(), std::regex_replace(), std::regex_replace(), std::regex_replace(), std::regex_replace(), std::regex_replace(), std::regex_search(), std::regex_search(), std::regex_search(), std::regex_search(), std::regex_search(), std::regex_search(), and std::regex_search().

match_not_bol

match_flag_type std::regex_constants::match_not_bol [inline], [constexpr]
The first character in the sequence [first, last) is treated as though it is not at the beginning of a line, so the character (^) in the regular expression shall not match [first, first).

match_not_bow

match_flag_type std::regex_constants::match_not_bow [inline], [constexpr]
The expression \b is not matched against the sub-sequence [first,first).

match_not_eol

match_flag_type std::regex_constants::match_not_eol [inline], [constexpr]
The last character in the sequence [first, last) is treated as though it is not at the end of a line, so the character ($) in the regular expression shall not match [last, last).

match_not_eow

match_flag_type std::regex_constants::match_not_eow [inline], [constexpr]
The expression \b should not be matched against the sub-sequence [last,last).

match_not_null

match_flag_type std::regex_constants::match_not_null [inline], [constexpr]
The expression does not match an empty sequence.
Referenced by std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++().

match_prev_avail

match_flag_type std::regex_constants::match_prev_avail [inline], [constexpr]
--first is a valid iterator position. When this flag is set then the flags match_not_bol and match_not_↩ bow are ignored by the algorithms regex_match, regex_search, and regex_replace, and by the iterators regex_iterator and regex_token_iterator.
Referenced by std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++().

multiline

`syntax_option_type std::regex_constants::multiline [inline], [constexpr]`
Specifies that the ^ anchor matches at the beginning of a line, and the $ anchor matches at the end of a line, not only at the beginning/end of the input. Valid for the ECMAScript syntax, ignored otherwise.

Since

C++17

nosubs

`syntax_option_type std::regex_constants::nosubs [inline], [constexpr]`
Specifies that when a regular expression is matched against a character container sequence, no sub-expression matches are to be stored in the supplied match_results structure.

optimize

`syntax_option_type std::regex_constants::optimize [inline], [constexpr]`
Specifies that the regular expression engine should pay more attention to the speed with which regular expressions are matched, and less to the speed with which regular expression objects are constructed. Otherwise it has no detectable effect on the program output.

4.22 std::rel_ops Namespace Reference

Functions

- template<class _Tp>
  bool operator!= (const _Tp &__x, const _Tp &__y)
- template<class _Tp>
  bool operator<= (const _Tp &__x, const _Tp &__y)
- template<class _Tp>
  bool operator> (const _Tp &__x, const _Tp &__y)
- template<class _Tp>
  bool operator>= (const _Tp &__x, const _Tp &__y)

4.22.1 Detailed Description

The generated relational operators are sequestered here.
Libstdc++ headers must not use the contents of `rel_ops`. User code should also avoid them, because unconstrained function templates are too greedy and can easily cause ambiguities.
C++20 default comparisons are a better solution.

4.22.2 Function Documentation

operator"!=()

```
template<class _Tp>
bool std::rel_ops::operator!= (
            const _Tp & __x,
            const _Tp & __y)  [inline]
```

Defines != for arbitrary types, in terms of ==.

Parameters

_↩ _x	A thing.
_↩ _y	Another thing.

Returns

__x != __y

This function uses == to determine its result.

operator<=()

```
template<class _Tp>
bool std::rel_ops::operator<= (
             const _Tp & __x,
             const _Tp & __y)  [inline]
```

Defines <= for arbitrary types, in terms of <.

Parameters

_↩ _x	A thing.
_↩ _y	Another thing.

Returns

__x <= __y

This function uses < to determine its result.

operator>()

```
template<class _Tp>
bool std::rel_ops::operator> (
             const _Tp & __x,
             const _Tp & __y)  [inline]
```

Defines > for arbitrary types, in terms of <.

Parameters

_↩ _x	A thing.
_↩ _y	Another thing.

Returns

__x > __y

This function uses < to determine its result.

operator>=()

```
template<class _Tp>
bool std::rel_ops::operator>= (
            const _Tp & __x,
            const _Tp & __y)  [inline]
```

Defines >= for arbitrary types, in terms of <.

Parameters

_↩ _x	A thing.
_↩ _y	Another thing.

Returns

__x >= __y

This function uses < to determine its result.

4.23 std::this_thread Namespace Reference

Functions

- thread::id get_id () noexcept
- template<typename _Rep, typename _Period>
  void sleep_for (const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock, typename _Duration>
  void sleep_until (const chrono::time_point< _Clock, _Duration > &__atime)
- void yield () noexcept

4.23.1 Detailed Description

ISO C++ 2011 namespace for interacting with the current thread.
C++11 30.3.2 [thread.thread.this] Namespace this_thread.

4.23.2 Function Documentation

get_id()

```
thread::id std::this_thread::get_id ()  [inline], [noexcept]
```

The unique identifier of the current thread.

sleep_for()

```
template<typename _Rep, typename _Period>
void std::this_thread::sleep_for (
            const chrono::duration< _Rep, _Period > & __rtime)  [inline]
```

this_thread::sleep_for
Referenced by sleep_until().

sleep_until()

```
template<typename _Clock, typename _Duration>
void std::this_thread::sleep_until (
            const chrono::time_point< _Clock, _Duration > & __atime)  [inline]
```

this_thread::sleep_until
References sleep_for().

yield()

```
void std::this_thread::yield ()  [inline], [noexcept]
```

Allow the implementation to schedule a different thread.

4.24 std::tr1 Namespace Reference

Namespaces

- namespace __detail

Functions

- template<typename _Tp>
  std::complex< _Tp > **acos** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **acosh** (const std::complex< _Tp > &)
- template<typename _Tp>
  _Tp arg (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **asin** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **asinh** (const std::complex< _Tp > &)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float **assoc_laguerref** (unsigned int __n, unsigned int __m, float __x)
- long double **assoc_laguerrel** (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float **assoc_legendref** (unsigned int __l, unsigned int __m, float __x)
- long double **assoc_legendrel** (unsigned int __l, unsigned int __m, long double __x)
- template<typename _Tp>
  std::complex< _Tp > **atan** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **atanh** (const std::complex< _Tp > &)
- template<typename _Tpx, typename _Tpy>
  __gnu_cxx::__promote_2< _Tpx, _Tpy >::__type beta (_Tpx __x, _Tpy __y)
- float **betaf** (float __x, float __y)
- long double **betal** (long double __x, long double __y)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type comp_ellint_1 (_Tp __k)
- float **comp_ellint_1f** (float __k)
- long double **comp_ellint_1l** (long double __k)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type comp_ellint_2 (_Tp __k)
- float **comp_ellint_2f** (float __k)

- long double **comp_ellint_2l** (long double __k)
- template<typename _Tp, typename _Tpn>
  __gnu_cxx::__promote_2< _Tp, _Tpn >::__type comp_ellint_3 (_Tp __k, _Tpn __nu)
- float **comp_ellint_3f** (float __k, float __nu)
- long double **comp_ellint_3l** (long double __k, long double __nu)
- template<typename _Tpa, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)
- float conf_hypergf (float __a, float __c, float __x)
- long double conf_hypergl (long double __a, long double __c, long double __x)
- template<typename _Tp>
  std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **conj** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **conj** (const std::complex< _Tp > &__z)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float **cyl_bessel_if** (float __nu, float __x)
- long double **cyl_bessel_il** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float **cyl_bessel_jf** (float __nu, float __x)
- long double **cyl_bessel_jl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float **cyl_bessel_kf** (float __nu, float __x)
- long double **cyl_bessel_kl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type cyl_neumann (_Tpnu __nu, _Tp __x)
- float **cyl_neumannf** (float __nu, float __x)
- long double **cyl_neumannl** (long double __nu, long double __x)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_1 (_Tp __k, _Tpp __phi)
- float **ellint_1f** (float __k, float __phi)
- long double **ellint_1l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type ellint_2 (_Tp __k, _Tpp __phi)
- float **ellint_2f** (float __k, float __phi)
- long double **ellint_2l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp>
  __gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float **ellint_3f** (float __k, float __nu, float __phi)
- long double **ellint_3l** (long double __k, long double __nu, long double __phi)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type expint (_Tp __x)
- float **expintf** (float __x)
- long double **expintl** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type **fabs** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > fabs (const std::complex< _Tp > &__z)
- float **fabs** (float __x)
- long double **fabs** (long double __x)

- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type hermite (unsigned int __n, _Tp __x)
- float **hermitef** (unsigned int __n, float __x)
- long double **hermitel** (unsigned int __n, long double __x)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float hypergf (float __a, float __b, float __c, float __x)
- long double hypergl (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp>
  constexpr _Tp **imag** (const complex< _Tp > &__z)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type laguerre (unsigned int __n, _Tp __x)
- float **laguerref** (unsigned int __n, float __x)
- long double **laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type legendre (unsigned int __n, _Tp __x)
- float **legendref** (unsigned int __n, float __x)
- long double **legendrel** (unsigned int __n, long double __x)
- template<typename _Tp>
  _Tp constexpr norm (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > polar (const _Tp &, const _Tp &=_Tp(0))
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **polar** (const _Tp &__rho, const _Up &__theta)
- template<typename _Tp, typename _Up>
  __gnu_cxx::__promote_2< _Tp, _Up >::__type **pow** (_Tp __x, _Up __y)
- template<typename _Tp>
  std::complex< _Tp > **pow** (const _Tp &__x, const std::complex< _Tp > &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **pow** (const _Tp &__x, const std::complex< _Up > &__y)
- template<typename _Tp>
  std::complex< _Tp > **pow** (const std::complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > pow (const std::complex< _Tp > &__x, const _Up &__y)
- template<typename _Tp>
  std::complex< _Tp > **pow** (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)
- float **pow** (float __x, float __y)
- long double **pow** (long double __x, long double __y)
- template<typename _Tp>
  constexpr _Tp **real** (const complex< _Tp > &__z)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type riemann_zeta (_Tp __x)
- float **riemann_zetaf** (float __x)
- long double **riemann_zetal** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type sph_bessel (unsigned int __n, _Tp __x)

- float **sph_besself** (unsigned int __n, float __x)
- long double **sph_bessell** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float **sph_legendref** (unsigned int __l, unsigned int __m, float __theta)
- long double **sph_legendrel** (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type sph_neumann (unsigned int __n, _Tp __x)
- float **sph_neumannf** (unsigned int __n, float __x)
- long double **sph_neumannl** (unsigned int __n, long double __x)

4.24.1 Detailed Description

ISO C++ TR1 entities toplevel namespace is std::tr1.

4.24.2 Function Documentation

arg()

```
template<typename _Tp>
_Tp std::arg (
             const complex< _Tp > & __z)  [inline]
```

Return phase angle of *z*.

conf_hyperg()

```
template<typename _Tpa, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (
             _Tpa __a,
             _Tpc __c,
             _Tp __x)  [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a;c;x)$ of real numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.
The confluent hypergeometric function is defined by

$$ {}_1F_1(a;c;x) = \sum_{n=0}^{\infty} \frac{(a)_n x^n}{(c)_n n!} $$

where the Pochhammer symbol is $(x)_k = (x)(x+1)...(x+k-1)$, $(x)_0 = 1$

Parameters

_↩ _*a*	The numeratorial parameter
_↩ _*c*	The denominatorial parameter
_↩ _*x*	The argument

conf_hypergf()

```
float __gnu_cxx::conf_hypergf (
            float __a,
            float __c,
            float __x)  [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a;c;x)$ of `float` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

See also

conf_hyperg for details.

conf_hypergl()

```
long double __gnu_cxx::conf_hypergl (
            long double __a,
            long double __c,
            long double __x)  [inline]
```

Return the confluent hypergeometric function ${}_1F_1(a;c;x)$ of `long double` numeratorial parameter `a`, denominatorial parameter `c`, and argument `x`.

See also

conf_hyperg for details.

hyperg()

```
template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
__gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg (
            _Tpa __a,
            _Tpb __b,
            _Tpc __c,
            _Tp __x)  [inline]
```

Return the hypergeometric function ${}_2F_1(a,b;c;x)$ of real numeratorial parameters `a` and `b`, denominatorial parameter `c`, and argument `x`.
The hypergeometric function is defined by

$$ {}_2F_1(a;c;x) = \sum_{n=0}^{\infty} \frac{(a)_n (b)_n x^n}{(c)_n n!} $$

where the Pochhammer symbol is $(x)_k = (x)(x+1)...(x+k-1)$, $(x)_0 = 1$

Parameters

_↩ _a	The first numeratorial parameter
_↩ _b	The second numeratorial parameter
_↩ _c	The denominatorial parameter
_↩ _x	The argument

hypergf()

```
float __gnu_cxx::hypergf (
            float __a,
            float __b,
            float __c,
            float __x)  [inline]
```

Return the hypergeometric function ${}_2F_1(a,b;c;x)$ of @ float numeratorial parameters a and b, denominatorial parameter c, and argument x.

See also

hyperg for details.

hypergl()

```
long double __gnu_cxx::hypergl (
            long double __a,
            long double __b,
            long double __c,
            long double __x)  [inline]
```

Return the hypergeometric function ${}_2F_1(a,b;c;x)$ of long double numeratorial parameters a and b, denominatorial parameter c, and argument x.

See also

hyperg for details.

norm()

```
template<typename _Tp>
_Tp constexpr std::norm (
            const complex< _Tp > & )  [constexpr]
```

Return *z* magnitude squared.

polar()

```
template<typename _Tp>
complex< _Tp > std::polar (
            const _Tp & __rho,
            const _Tp & __theta = _Tp(0))  [inline]
```

Return complex with magnitude *rho* and angle *theta*.

4.25 std::tr1::__detail Namespace Reference

4.25.1 Detailed Description

Implementation details not part of the namespace std::tr1 interface.

4.26 std::tr2 Namespace Reference

Namespaces

- namespace __detail

Classes

- struct __dynamic_bitset_base
- struct __reflection_typelist
- struct __reflection_typelist< _First, _Rest... >
- struct __reflection_typelist<>
- struct bases
- class bool_set
- struct direct_bases
- class dynamic_bitset

Functions

- bool **certainly** (bool_set __b)
- bool **contains** (bool_set __s, bool_set __t)
- bool **equals** (bool_set __s, bool_set __t)
- bool **is_emptyset** (bool_set __b)
- bool **is_indeterminate** (bool_set __b)
- bool **is_singleton** (bool_set __b)
- bool_set **operator!=** (bool __s, bool_set __t)
- bool_set **operator!=** (bool_set __s, bool __t)
- bool_set **operator!=** (bool_set __s, bool_set __t)
- bool_set **operator&** (bool __s, bool_set __t)
- bool_set **operator&** (bool_set __s, bool __t)
- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const dynamic_bitset< _WordT, _Alloc > &__x)
- bool_set **operator==** (bool __s, bool_set __t)
- bool_set **operator==** (bool_set __s, bool __t)
- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, dynamic_bitset< _WordT, _Alloc > &__x)
- bool_set **operator^** (bool __s, bool_set __t)
- bool_set **operator^** (bool_set __s, bool __t)
- bool_set **operator|** (bool __s, bool_set __t)
- bool_set **operator|** (bool_set __s, bool __t)
- bool **possibly** (bool_set __b)
- bool_set **set_complement** (bool_set __b)
- bool_set **set_intersection** (bool __s, bool_set __t)
- bool_set **set_intersection** (bool_set __s, bool __t)
- bool_set **set_intersection** (bool_set __s, bool_set __t)
- bool_set **set_union** (bool __s, bool_set __t)
- bool_set **set_union** (bool_set __s, bool __t)
- bool_set **set_union** (bool_set __s, bool_set __t)

- template<typename _WordT, typename _Alloc>
  bool operator!= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool operator<= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)

- template<typename _WordT, typename _Alloc>
  bool operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)

- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)

4.26.1 Detailed Description

Namespace for non-standard "TR2" extensions.

4.27 std::tr2::__detail Namespace Reference

4.27.1 Detailed Description

Implementation details not part of the namespace std::tr2 interface.

5 Concept Documentation

5.1 std::assignable_from Concept Reference

`#include <concepts>`

5.1.1 Concept definition

```
template<typename _Lhs, typename _Rhs>
concept assignable_from = is_lvalue_reference_v<_Lhs>
  && common_reference_with<__detail::__cref<_Lhs>, __detail::__cref<_Rhs»
  && requires(_Lhs __lhs, _Rhs&& __rhs) {
    { __lhs = static_cast<_Rhs&&>(__rhs) } -> same_as<_Lhs>;
  }
```

5.1.2 Detailed Description

[concept.assignable], concept assignable_from

5.2 std::common_reference_with Concept Reference

`#include <concepts>`

5.2.1 Concept definition

```
template<typename _Tp, typename _Up>
concept common_reference_with = same_as<common_reference_t<_Tp, _Up>, common_reference_t<_Up, _Tp>>
  && convertible_to<_Tp, common_reference_t<_Tp, _Up>>
  && convertible_to<_Up, common_reference_t<_Tp, _Up>>
```

5.2.2 Detailed Description

[concept.commonref], concept common_reference_with

5.3 std::common_with Concept Reference

```
#include <concepts>
```

5.3.1 Concept definition

```
template<typename _Tp, typename _Up>
concept common_with = same_as<common_type_t<_Tp, _Up>, common_type_t<_Up, _Tp>>
  && requires {
    static_cast<common_type_t<_Tp, _Up>>(std::declval<_Tp>());
    static_cast<common_type_t<_Tp, _Up>>(std::declval<_Up>());
  }
  && common_reference_with<add_lvalue_reference_t<const _Tp>,
                           add_lvalue_reference_t<const _Up>>
  && common_reference_with<add_lvalue_reference_t<common_type_t<_Tp, _Up>>,
                           common_reference_t<
                             add_lvalue_reference_t<const _Tp>,
                             add_lvalue_reference_t<const _Up>»
```

5.3.2 Detailed Description

[concept.common], concept common_with

5.4 std::constructible_from Concept Reference

```
#include <concepts>
```

5.4.1 Concept definition

```
template<typename _Tp, typename... _Args>
concept constructible_from = destructible<_Tp> && is_constructible_v<_Tp, _Args...>
```

5.4.2 Detailed Description

[concept.constructible], concept constructible_from

5.5 std::convertible_to Concept Reference

```
#include <concepts>
```

5.5.1 Concept definition

```
template<typename _From, typename _To>
concept convertible_to = is_convertible_v<_From, _To>
  && requires { static_cast<_To>(std::declval<_From>()); }
```

5.5.2 Detailed Description

[concept.convertible], concept convertible_to

5.6 std::copy_constructible Concept Reference

`#include <concepts>`

5.6.1 Concept definition

```
template<typename _Tp>
concept copy_constructible = move_constructible<_Tp>
  && constructible_from<_Tp, _Tp&> && convertible_to<_Tp&, _Tp>
  && constructible_from<_Tp, const _Tp&> && convertible_to<const _Tp&, _Tp>
  && constructible_from<_Tp, const _Tp> && convertible_to<const _Tp, _Tp>
```

5.6.2 Detailed Description

[concept.copyconstructible], concept copy_constructible

5.7 std::default_initializable Concept Reference

`#include <concepts>`

5.7.1 Concept definition

```
template<typename _Tp>
concept default_initializable = constructible_from<_Tp>
  && requires
  {
    _Tp{};
    (void) ::new _Tp;
  }
```

5.7.2 Detailed Description

[concept.defaultinitializable], concept default_initializable

5.8 std::derived_from Concept Reference

`#include <concepts>`

5.8.1 Concept definition

```
template<typename _Derived, typename _Base>
concept derived_from = __is_base_of(_Base, _Derived)
  && is_convertible_v<const volatile _Derived*, const volatile _Base*>
```

5.8.2 Detailed Description

[concept.derived], concept derived_from

5.9 std::destructible Concept Reference

`#include <concepts>`

5.9.1 Concept definition

```
template<typename _Tp>
concept destructible = __detail::__destructible<_Tp>
```

5.9.2 Detailed Description

[concept.destructible], concept destructible

5.10 std::equivalence_relation Concept Reference

`#include <concepts>`

5.10.1 Concept definition

```
template<typename _Rel, typename _Tp, typename _Up>
concept equivalence_relation = relation<_Rel, _Tp, _Up>
```

5.10.2 Detailed Description

[concept.equiv], concept equivalence_relation

5.11 std::indirectly_comparable Concept Reference

`#include <iterator_concepts.h>`

5.11.1 Concept definition

```
template<typename _I1, typename _I2, typename _Rel, typename _P1 = identity, typename _P2 = identity>
concept indirectly_comparable = indirect_binary_predicate<_Rel, projected<_I1, _P1>,
                                projected<_I2, _P2>>
```

5.11.2 Detailed Description

[alg.req.ind.cmp], concept `indirectly_comparable`

5.12 std::indirectly_copyable Concept Reference

`#include <iterator_concepts.h>`

5.12.1 Concept definition

```
template<typename _In, typename _Out>
concept indirectly_copyable = indirectly_readable<_In>
  && indirectly_writable<_Out, iter_reference_t<_In>>
```

5.12.2 Detailed Description

[alg.req.ind.copy], concept `indirectly_copyable`

5.13 std::indirectly_movable Concept Reference

`#include <iterator_concepts.h>`

5.13.1 Concept definition

```
template<typename _In, typename _Out>
concept indirectly_movable = indirectly_readable<_In>
  && indirectly_writable<_Out, iter_rvalue_reference_t<_In>>
```

5.13.2 Detailed Description

[alg.req.ind.move], concept `indirectly_movable`

5.14 std::indirectly_readable Concept Reference

`#include <iterator_concepts.h>`

5.14.1 Concept definition

```
template<typename _In>
concept indirectly_readable = __detail::__indirectly_readable_impl<remove_cvref_t<_In»
```

5.14.2 Detailed Description

Requirements for types that are readable by applying operator∗.

5.15 std::indirectly_swappable Concept Reference

`#include <iterator_concepts.h>`

5.15.1 Concept definition

```
template<typename _I1, typename _I2 = _I1>
concept indirectly_swappable = indirectly_readable<_I1> && indirectly_readable<_I2>
  && requires(const _I1 __i1, const _I2 __i2)
  {
    ranges::iter_swap(__i1, __i1);
    ranges::iter_swap(__i2, __i2);
    ranges::iter_swap(__i1, __i2);
    ranges::iter_swap(__i2, __i1);
  }
```

5.15.2 Detailed Description

[alg.req.ind.swap], concept `indirectly_swappable`

5.16 std::indirectly_writable Concept Reference

`#include <iterator_concepts.h>`

5.16.1 Concept definition

```
template<typename _Out, typename _Tp>
concept indirectly_writable = requires(_Out&& __o, _Tp&& __t)
  {
    *__o = std::forward<_Tp>(__t);
    *std::forward<_Out>(__o) = std::forward<_Tp>(__t);
    const_cast<const iter_reference_t<_Out>&&>(*__o)
      = std::forward<_Tp>(__t);
    const_cast<const iter_reference_t<_Out>&&>(*std::forward<_Out>(__o))
      = std::forward<_Tp>(__t);
  }
```

5.16.2 Detailed Description

Requirements for writing a value into an iterator's referenced object.

5.17 std::invocable Concept Reference

`#include <concepts>`

5.17.1 Concept definition

```
template<typename _Fn, typename... _Args>
concept invocable = is_invocable_v<_Fn, _Args...>
```

5.17.2 Detailed Description

[concept.invocable], concept invocable

5.18 std::mergeable Concept Reference

`#include <iterator_concepts.h>`

5.18.1 Concept definition

```
template<typename _I1, typename _I2, typename _Out, typename _Rel = ranges::less, typename _P1 = identity,
  typename _P2 = identity>
concept mergeable = input_iterator<_I1> && input_iterator<_I2>
  && weakly_incrementable<_Out> && indirectly_copyable<_I1, _Out>
  && indirectly_copyable<_I2, _Out>
  && indirect_strict_weak_order<_Rel, projected<_I1, _P1>,
                                projected<_I2, _P2>>
```

5.18.2 Detailed Description

[alg.req.mergeable], concept `mergeable`

5.19 std::move_constructible Concept Reference

`#include <concepts>`

5.19.1 Concept definition

```
template<typename _Tp>
concept move_constructible = constructible_from<_Tp, _Tp> && convertible_to<_Tp, _Tp>
```

5.19.2 Detailed Description

[concept.moveconstructible], concept move_constructible

5.20 std::permutable Concept Reference

`#include <iterator_concepts.h>`

5.20.1 Concept definition

```
template<typename _Iter>
concept permutable = forward_iterator<_Iter>
  && indirectly_movable_storable<_Iter, _Iter>
  && indirectly_swappable<_Iter, _Iter>
```

5.20.2 Detailed Description

[alg.req.permutable], concept `permutable`

5.21 std::predicate Concept Reference

`#include <concepts>`

5.21.1 Concept definition

```
template<typename _Fn, typename... _Args>
concept predicate = regular_invocable<_Fn, _Args...>
  && __detail::__boolean_testable<invoke_result_t<_Fn, _Args...»
```

5.21.2 Detailed Description

[concept.predicate], concept predicate

5.22 std::ranges::bidirectional_range Concept Reference

`#include <ranges_base.h>`

5.22.1 Concept definition

```
template<typename _Tp>
concept bidirectional_range = forward_range<_Tp> && bidirectional_iterator<iterator_t<_Tp»
```

5.22.2 Detailed Description

A range for which ranges::begin returns a bidirectional iterator.

5.23 std::ranges::borrowed_range Concept Reference

`#include <ranges_base.h>`

5.23.1 Concept definition

```
template<typename _Tp>
concept borrowed_range = range<_Tp> && __detail::__maybe_borrowed_range<_Tp>
```

5.23.2 Detailed Description

[range.range] The borrowed_range concept.

5.24 std::ranges::common_range Concept Reference

`#include <ranges_base.h>`

5.24.1 Concept definition

```
template<typename _Tp>
concept common_range = range<_Tp> && same_as<iterator_t<_Tp>, sentinel_t<_Tp»
```

5.24.2 Detailed Description

A range for which ranges::begin and ranges::end return the same type.

5.25 std::ranges::contiguous_range Concept Reference

`#include <ranges_base.h>`

5.25.1 Concept definition

```
template<typename _Tp>
concept contiguous_range = random_access_range<_Tp> && contiguous_iterator<iterator_t<_Tp»
  && requires(_Tp& __t)
  {
    { ranges::data(__t) } -> same_as<add_pointer_t<range_reference_t<_Tp>»;
  }
```

5.25.2 Detailed Description

A range for which ranges::begin returns a contiguous iterator.

5.26 std::ranges::forward_range Concept Reference

`#include <ranges_base.h>`

5.26.1 Concept definition

```
template<typename _Tp>
concept forward_range = input_range<_Tp> && forward_iterator<iterator_t<_Tp»
```

5.26.2 Detailed Description

A range for which ranges::begin returns a forward iterator.

5.27 std::ranges::input_range Concept Reference

`#include <ranges_base.h>`

5.27.1 Concept definition

```
template<typename _Tp>
concept input_range = range<_Tp> && input_iterator<iterator_t<_Tp»
```

5.27.2 Detailed Description

A range for which ranges::begin returns an input iterator.

5.28 std::ranges::output_range Concept Reference

`#include <ranges_base.h>`

5.28.1 Concept definition

```
template<typename _Range, typename _Tp>
concept output_range = range<_Range> && output_iterator<iterator_t<_Range>, _Tp>
```

5.28.2 Detailed Description

A range for which ranges::begin returns an output iterator.

5.29 std::ranges::random_access_range Concept Reference

`#include <ranges_base.h>`

5.29.1 Concept definition

```
template<typename _Tp>
concept random_access_range = bidirectional_range<_Tp> && random_access_iterator<iterator_t<_Tp»
```

5.29.2 Detailed Description

A range for which ranges::begin returns a random access iterator.

5.30 std::ranges::range Concept Reference

`#include <ranges_base.h>`

5.30.1 Concept definition

```
template<typename _Tp>
concept range = requires(_Tp& __t)
  {
    ranges::begin(__t);
    ranges::end(__t);
  }
```

5.30.2 Detailed Description

[range.range] The range concept.

5.31 std::ranges::sized_range Concept Reference

`#include <ranges_base.h>`

5.31.1 Concept definition

```
template<typename _Tp>
concept sized_range = range<_Tp>
  && requires(_Tp& __t) { ranges::size(__t); }
```

5.31.2 Detailed Description

[range.sized] The sized_range concept.

5.32 std::ranges::view Concept Reference

`#include <ranges_base.h>`

5.32.1 Concept definition

```
template<typename _Tp>
concept view = range<_Tp> && movable<_Tp> && enable_view<_Tp>
```

5.32.2 Detailed Description

[range.view] The ranges::view concept.

5.33 std::ranges::viewable_range Concept Reference

`#include <ranges_base.h>`

5.33.1 Concept definition

```
template<typename _Tp>
concept viewable_range = range<_Tp>
  && ((view<remove_cvref_t<_Tp>> && constructible_from<remove_cvref_t<_Tp>, _Tp>)
      || (!view<remove_cvref_t<_Tp>>
          && (is_lvalue_reference_v<_Tp>
              || (movable<remove_reference_t<_Tp»
                  && !__detail::__is_initializer_list<remove_cvref_t<_Tp»))))
```

5.33.2 Detailed Description

A range which can be safely converted to a view.

5.34 std::regular_invocable Concept Reference

`#include <concepts>`

5.34.1 Concept definition

```
template<typename _Fn, typename... _Args>
concept regular_invocable = invocable<_Fn, _Args...>
```

5.34.2 Detailed Description

[concept.regularinvocable], concept regular_invocable

5.35 std::relation Concept Reference

`#include <concepts>`

5.35.1 Concept definition

```
template<typename _Rel, typename _Tp, typename _Up>
concept relation = predicate<_Rel, _Tp, _Tp> && predicate<_Rel, _Up, _Up>
  && predicate<_Rel, _Tp, _Up> && predicate<_Rel, _Up, _Tp>
```

5.35.2 Detailed Description

[concept.relation], concept relation

5.36 std::same_as Concept Reference

`#include <concepts>`

5.36.1 Concept definition

```
template<typename _Tp, typename _Up>
concept same_as = __detail::__same_as<_Tp, _Up> && __detail::__same_as<_Up, _Tp>
```

5.36.2 Detailed Description

[concept.same], concept same_as

5.37 std::simd::__vec_builtin Concept Reference

`#include <vec_ops.h>`

5.37.1 Concept definition

```
template<typename _Tp>
concept __vec_builtin = __vec_builtin_of<_Tp, remove_cvref_t<decltype(declval<const _Tp>()[0])»
```

5.37.2 Detailed Description

Constrain to any vector builtin.

5.38 std::simd::__vec_builtin_of Concept Reference

`#include <vec_ops.h>`

5.38.1 Concept definition

```
template<typename _Tp, typename _ValueType, __simd_size_type _Width = sizeof(_Tp) / sizeof(_ValueType)>
concept __vec_builtin_of = !is_class_v<_Tp> && !is_pointer_v<_Tp> && !is_arithmetic_v<_Tp>
      && __vectorizable<_ValueType>
      && _Width >= 1 && sizeof(_Tp) / sizeof(_ValueType) == _Width
      && same_as<__vec_builtin_type_bytes<_ValueType, sizeof(_Tp)>, _Tp>
      && requires(_Tp& __v, _ValueType __x) { __v[0] = __x; }
```

5.38.2 Detailed Description

Constrain to any vector builtin with given value type and optional width.

5.39 std::sortable Concept Reference

`#include <iterator_concepts.h>`

5.39.1 Concept definition

```
template<typename _Iter, typename _Rel = ranges::less, typename _Proj = identity>
concept sortable = permutable<_Iter>
  && indirect_strict_weak_order<_Rel, projected<_Iter, _Proj»
```

5.39.2 Detailed Description

[alg.req.sortable], concept `sortable`

5.40 std::strict_weak_order Concept Reference

`#include <concepts>`

5.40.1 Concept definition

```
template<typename _Rel, typename _Tp, typename _Up>
concept strict_weak_order = relation<_Rel, _Tp, _Up>
```

5.40.2 Detailed Description

[concept.strictweakorder], concept strict_weak_order

5.41 std::uniform_random_bit_generator Concept Reference

`#include <random>`

5.41.1 Concept definition

```
template<typename _Gen>
concept uniform_random_bit_generator = invocable<_Gen&> && unsigned_integral<invoke_result_t<_Gen&»
  && requires
  {
    { _Gen::min() } -> same_as<invoke_result_t<_Gen&>>;
    { _Gen::max() } -> same_as<invoke_result_t<_Gen&»;
    requires bool_constant<(_Gen::min() < _Gen::max())>::value;
  }
```

5.41.2 Detailed Description

Requirements for a uniform random bit generator.

Since

C++20

5.42 std::weakly_incrementable Concept Reference

`#include <iterator_concepts.h>`

5.42.1 Concept definition

```
template<typename _Iter>
concept weakly_incrementable = movable<_Iter>
  && requires(_Iter __i)
  {
    typename iter_difference_t<_Iter>;
    requires __detail::__is_signed_integer_like<iter_difference_t<_Iter»;
    { ++__i } -> same_as<_Iter&>;
    __i++;
  }
```

5.42.2 Detailed Description

Requirements on types that can be incremented with ++.

6 Class Documentation

6.1 __gnu_parallel::__accumulate_binop_reduct< _BinOp > Struct Template Reference

`#include <for_each_selectors.h>`

Public Member Functions

- **__accumulate_binop_reduct** (_BinOp &__b)
- template<typename _Result, typename _Addend>
  _Result **operator()** (const _Result &__x, const _Addend &__y)

Public Attributes

- _BinOp & **__binop**

6.1.1 Detailed Description

template<typename _BinOp>
struct __gnu_parallel::__accumulate_binop_reduct< _BinOp >

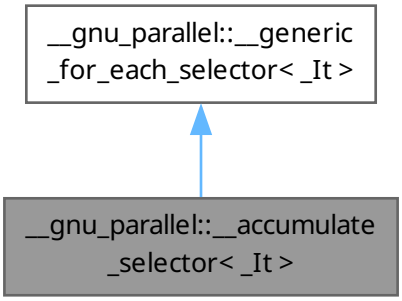
General reduction, using a binary operator.
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.2 __gnu_parallel::__accumulate_selector< _It > Struct Template Reference

`#include <for_each_selectors.h>`

Inheritance diagram for __gnu_parallel::__accumulate_selector< _It >:



Public Member Functions

- template<typename _Op>
  std::iterator_traits< _It >::value_type operator() (_Op __o, _It __i)

Public Attributes

- _It _M_finish_iterator

6.2.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__accumulate_selector< _It >

std::accumulate() selector.

6.2.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _Op>
std::iterator_traits< _It >::value_type __gnu_parallel::__accumulate_selector< _It >::operator()
(
            _Op __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator (unused).

_↩ _i	iterator referencing object.

Returns

The current value.

6.2.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

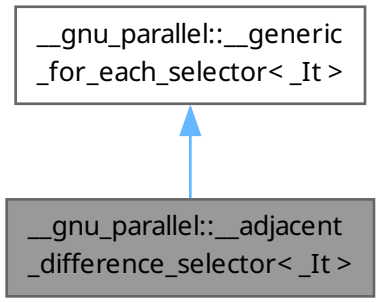
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.3 __gnu_parallel::__adjacent_difference_selector< _It > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__adjacent_difference_selector< _It >:



Public Member Functions

- template<typename _Op>
  bool **operator()** (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

6.3.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__adjacent_difference_selector< _It >

Selector that returns the difference between two adjacent __elements.

6.3.2 Member Data Documentation

_M_finish_iterator

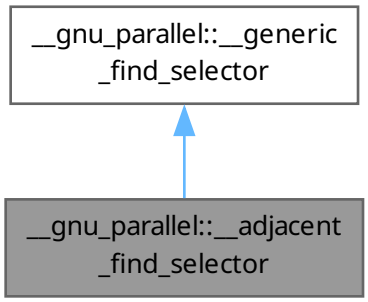
```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.4 __gnu_parallel::__adjacent_find_selector Struct Reference

`#include <find_selectors.h>`
Inheritance diagram for __gnu_parallel::__adjacent_find_selector:



Public Member Functions

- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __↩
  begin2, _Pred __pred)
- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)

6.4.1 Detailed Description

Test predicate on two adjacent elements.

6.4.2 Member Function Documentation

_M_sequential_algorithm()

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__adjacent_find_selector::_M_sequential_algorithm
(
            _RAIter1 __begin1,
            _RAIter1 __end1,
            _RAIter2 __begin2,
            _Pred __pred)  [inline]
```

Corresponding sequential algorithm on a sequence.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__pred	Find predicate.

References std::make_pair().

operator()()

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__adjacent_find_selector::operator() (
            _RAIter1 __i1,
            _RAIter2 __i2,
            _Pred __pred)  [inline]
```

Test on one position.

Parameters

__i1	_Iterator on first sequence.
__i2	_Iterator on second sequence (unused).
__pred	Find predicate.

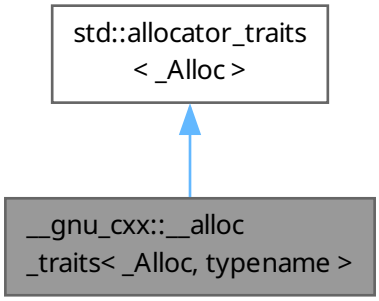
The documentation for this struct was generated from the following file:

- find_selectors.h

6.5 __gnu_cxx::__alloc_traits< _Alloc, typename > Struct Template Reference

```
#include <alloc_traits.h>
```

Inheritance diagram for __gnu_cxx::__alloc_traits< _Alloc, typename >:



Public Types

- typedef std::allocator_traits< _Alloc > **_Base_type**
- typedef _Alloc **allocator_type**
- typedef _Base_type::const_pointer **const_pointer**
- typedef const value_type & **const_reference**
- using const_void_pointer
- typedef _Base_type::difference_type **difference_type**
- using is_always_equal
- typedef _Base_type::pointer **pointer**
- using propagate_on_container_copy_assignment
- using propagate_on_container_move_assignment
- using propagate_on_container_swap
- template<typename _Tp>
  using **rebind_alloc**
- template<typename _Tp>
  using **rebind_traits**
- typedef value_type & **reference**
- typedef _Base_type::size_type **size_type**
- typedef _Base_type::value_type **value_type**
- using void_pointer

Static Public Member Functions

- static constexpr bool **_S_always_equal** ()
- static constexpr bool **_S_nothrow_move** ()
- static constexpr void **_S_on_swap** (_Alloc &__a, _Alloc &__b)
- static constexpr bool **_S_propagate_on_copy_assign** ()
- static constexpr bool **_S_propagate_on_move_assign** ()
- static constexpr bool **_S_propagate_on_swap** ()
- static constexpr _Alloc **_S_select_on_copy** (const _Alloc &__a)
- static constexpr pointer allocate (_Alloc &__a, size_type __n)
- static constexpr pointer allocate (_Alloc &__a, size_type __n, const_void_pointer __hint)

- static constexpr pointer allocate (_Alloc &__a, size_type __n)
- static constexpr pointer allocate (_Alloc &__a, size_type __n, const_void_pointer __hint)
- static constexpr allocation_result< pointer, size_type > allocate_at_least (_Alloc &__a, size_type __n)
- template<typename _Ptr, typename... _Args>
  static constexpr std::__enable_if_t< __is_custom_pointer< _Ptr >::value > **construct** (_Alloc &__a, _Ptr _↩
  _p, _Args &&... __args) noexcept(noexcept(_Base_type::construct(__a, std::__to_address(__p), std::forward< _Args >(__args)...)))
- template<typename _Tp, typename... _Args>
  static constexpr void construct (_Alloc &__a, _Tp *__p, _Args &&... __args) noexcept(_S_nothrow_construct< _Tp, _Args... >())
- static constexpr void deallocate (_Alloc &__a, pointer __p, size_type __n)
- static constexpr void deallocate (_Alloc &__a, pointer __p, size_type __n)
- template<typename _Ptr>
  static constexpr std::__enable_if_t< __is_custom_pointer< _Ptr >::value > **destroy** (_Alloc &__a, _Ptr __p) noexcept(noexcept(_Base_type::destroy(__a, std::__to_address(__p))))
- template<typename _Tp>
  static constexpr void destroy (_Alloc &__a, _Tp *__p) noexcept(_S_nothrow_destroy< _Tp >())
- static constexpr size_type max_size (const _Alloc &__a) noexcept
- static constexpr _Alloc select_on_container_copy_construction (const _Alloc &__rhs)

6.5.1 Detailed Description

template<typename _Alloc, typename = typename _Alloc::value_type>
struct __gnu_cxx::__alloc_traits< _Alloc, typename >

Uniform interface to C++98 and C++11 allocators.

6.5.2 Member Typedef Documentation

const_void_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_void_pointer  [inherited]
```

The allocator's const void pointer type.
`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::↩ rebind<const void>`

is_always_equal

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::is_always_equal  [inherited]
```

Whether all instances of the allocator type compare equal.
`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

propagate_on_container_copy_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_copy_assignment  [inherited]
```

How the allocator is propagated on copy assignment.
`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

propagate_on_container_move_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_move_assignment  [inherited]
```

How the allocator is propagated on move assignment.

`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

propagate_on_container_swap

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_swap  [inherited]
```

How the allocator is propagated on swap.

`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

void_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::void_pointer  [inherited]
```

The allocator's void pointer type.

`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

6.5.3 Member Function Documentation

allocate() [1/4]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
constexpr pointer std::allocator_traits< _Alloc >::allocate (
            _Alloc & __a,
            size_type __n)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.

Calls `a.allocate(n)`

allocate() [2/4]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
constexpr pointer std::allocator_traits< _Alloc >::allocate (
            _Alloc & __a,
            size_type __n,
            const_void_pointer __hint)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

__a	An allocator.

__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

allocate() [3/4]

```
template<typename _Alloc>
constexpr pointer std::allocator_traits< _Alloc >::allocate (
            _Alloc & __a,
            size_type __n)  [inline], [static], [nodiscard], [constexpr], [inherited]
```

Allocate memory.

Parameters

_↩ _a	An allocator.
_↩ _n	The number of objects to allocate space for.

Calls `a.allocate(n)`

allocate() [4/4]

```
template<typename _Alloc>
constexpr pointer std::allocator_traits< _Alloc >::allocate (
            _Alloc & __a,
            size_type __n,
            const_void_pointer __hint)  [inline], [static], [nodiscard], [constexpr], [inherited]
```

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

allocate_at_least()

```
template<typename _Alloc>
constexpr allocation_result< pointer, size_type > std::allocator_traits< _Alloc >::allocate_at↩
_least (
            _Alloc & __a,
            size_type __n)  [inline], [static], [nodiscard], [constexpr], [inherited]
```

Allocate memory, generously.

Parameters

_↩ _a	An allocator.
_↩ _n	The minimum number of objects to allocate space for.

Returns

Memory of suitable size and alignment for `n` or more contiguous objects of type `value_type`.

Returns `a.allocate_at_least(n)` if that expression is well-formed, else `{ a.allocate(n), n }`. When an allocator is obliged to reserve more space than required for the cited `n` objects, it may deliver the extra space to the caller.

construct()

```
template<typename _Alloc, typename = typename _Alloc::value_type>
template<typename _Tp, typename...  _Args>
constexpr void std::allocator_traits< _Alloc >::construct (
            _Alloc & __a,
            _Tp * __p,
            _Args &&...  __args)  [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Tp`.

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for Tp
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type *_Tp* at location *__p* from the arguments *__args*...

deallocate() [1/2]

```
template<typename _Alloc, typename = typename _Alloc::value_type>
constexpr void std::allocator_traits< _Alloc >::deallocate (
            _Alloc & __a,
            pointer __p,
            size_type __n)  [inline], [static], [constexpr]
```

Deallocate memory.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the memory to deallocate.
_↩ _n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

deallocate() [2/2]

```
template<typename _Alloc>
constexpr void std::allocator_traits< _Alloc >::deallocate (
            _Alloc & __a,
            pointer __p,
            size_type __n)  [inline], [static], [constexpr], [inherited]
```

Deallocate memory.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the memory to deallocate.
_↩ _n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

destroy()

```
template<typename _Alloc, typename = typename _Alloc::value_type>
template<typename _Tp>
constexpr void std::allocator_traits< _Alloc >::destroy (
            _Alloc & __a,
            _Tp * __p)  [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type *_Tp*.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

max_size()

```
template<typename _Alloc, typename = typename _Alloc::value_type>
```

```
constexpr size_type std::allocator_traits< _Alloc >::max_size (
            const _Alloc & __a)  [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

Parameters

_↩ _a	An allocator.

Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_↩ type>::max()`

select_on_container_copy_construction()

```
template<typename _Alloc>
constexpr _Alloc std::allocator_traits< _Alloc >::select_on_container_copy_construction (
            const _Alloc & __rhs)  [inline], [static], [constexpr], [inherited]
```

Obtain an allocator to use when copying a container.

Parameters

__rhs	An allocator.

Returns

`__rhs.select_on_container_copy_construction()` or *__rhs*

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns *__rhs*

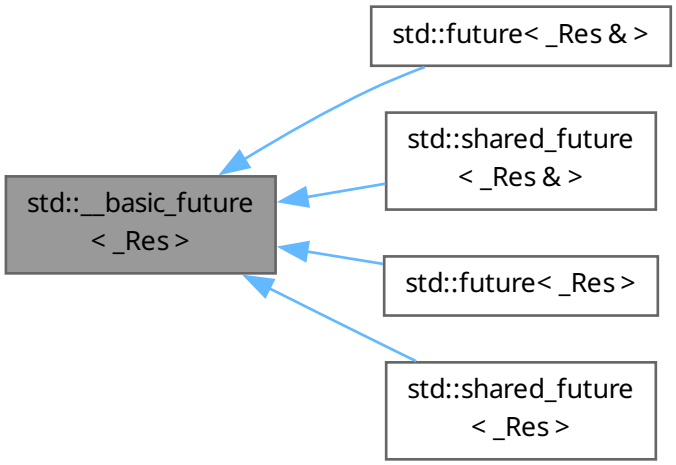
The documentation for this struct was generated from the following file:

- ext/alloc_traits.h

6.6 std::__basic_future< _Res > Class Template Reference

`#include <future>`

Inheritance diagram for std::__basic_future< _Res >:



Public Member Functions

- **__basic_future** (const __basic_future &)=delete
- __basic_future & **operator=** (const __basic_future &)=delete
- bool **valid** () const noexcept
- void **wait** () const
- template<typename _Rep, typename _Period>
  future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- template<typename _Clock, typename _Duration>
  future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< _Res > & **__result_type**
- typedef shared_ptr< _State_base > **__state_type**

Protected Member Functions

- **__basic_future** (const __state_type &__state)
- **__basic_future** (const shared_future< _Res > &) noexcept
- **__basic_future** (future< _Res > &&) noexcept
- **__basic_future** (shared_future< _Res > &&) noexcept
- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

6.6.1 Detailed Description

template<typename _Res>
class std::__basic_future< _Res >

Common implementation for future and shared_future.

6.6.2 Member Function Documentation

_M_get_result()

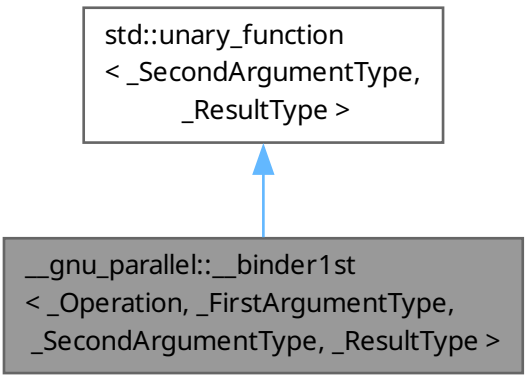
```
template<typename _Res>
__result_type std::__basic_future< _Res >::_M_get_result () const  [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.
Referenced by std::future< _Res >::get(), and std::shared_future< _Res >::get().
The documentation for this class was generated from the following file:

- future

6.7 __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference

`#include <base.h>`
Inheritance diagram for __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >:



Public Types

- typedef _SecondArgumentType argument_type
- typedef _ResultType result_type

Public Member Functions

- **__binder1st** (const _Operation &__x, const _FirstArgumentType &__y)
- _ResultType **operator()** (_SecondArgumentType &__x) const
- _ResultType **operator()** (const _SecondArgumentType &__x)

Protected Attributes

- _Operation **_M_op**
- _FirstArgumentType **_M_value**

6.7.1 Detailed Description

template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>
class __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >

Similar to std::binder1st, but giving the argument types explicitly.

6.7.2 Member Typedef Documentation

argument_type

```
typedef _SecondArgumentType std::unary_function< _SecondArgumentType, _ResultType >::argument_type [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _ResultType std::unary_function< _SecondArgumentType, _ResultType >::result_type [inherited]
```

`result_type` is the return type

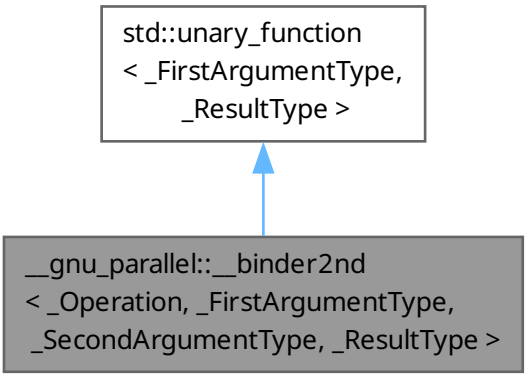
The documentation for this class was generated from the following file:

- base.h

6.8 __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType > Class Template Reference

```
#include <base.h>
```

Inheritance diagram for __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >:



Public Types

- typedef _FirstArgumentType argument_type
- typedef _ResultType result_type

Public Member Functions

- **__binder2nd** (const _Operation &__x, const _SecondArgumentType &__y)
- _ResultType **operator()** (_FirstArgumentType &__x)
- _ResultType **operator()** (const _FirstArgumentType &__x) const

Protected Attributes

- _Operation **_M_op**
- _SecondArgumentType **_M_value**

6.8.1 Detailed Description

template<typename _Operation, typename _FirstArgumentType, typename _SecondArgumentType, typename _ResultType>
class __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >

Similar to std::binder2nd, but giving the argument types explicitly.

6.8.2 Member Typedef Documentation

argument_type

```
typedef _FirstArgumentType std::unary_function< _FirstArgumentType, _ResultType >::argument_type
[inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _ResultType std::unary_function< _FirstArgumentType, _ResultType >::result_type  [inherited]
```

`result_type` is the return type

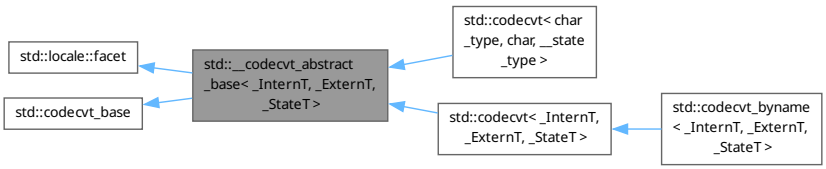
The documentation for this class was generated from the following file:

- base.h

6.9 std::__codecvt_abstract_base< _InternT, _ExternT, _StateT > Class Template Reference

`#include <codecvt.h>`

Inheritance diagram for std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >:



Public Types

- typedef _ExternT **extern_type**
- typedef _InternT **intern_type**
- typedef codecvt_base::result **result**
- typedef _StateT **state_type**

Public Member Functions

- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Protected Member Functions

- **__codecvt_abstract_base** (size_t __refs=0)
- virtual bool **do_always_noconv** () const throw () =0
- virtual int **do_encoding** () const throw () =0
- virtual result **do_in** (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const =0
- virtual int **do_length** (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const =0
- virtual int **do_max_length** () const throw () =0
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const =0
- virtual result **do_unshift** (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to↩ _next) const =0

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.9.1 Detailed Description

template<typename _InternT, typename _ExternT, typename _StateT>
class std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >

Common base for codecvt functions.
This template class provides implementations of the public functions that forward to the protected virtual functions.
This template also provides abstract stubs for the protected virtual functions.

6.9.2 Member Function Documentation

do_out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
```

```
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [pure virtual]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implemented in std::codecvt< _InternT, _ExternT, _StateT >, std::codecvt< _Elem, char, mbstate_t >, std::codecvt< _InternT, _ExternT
std::codecvt< char, char, mbstate_t >, std::codecvt< char16_t, char, mbstate_t >, std::codecvt< char32_t, char, mbstate_t >,
and std::codecvt< wchar_t, char, mbstate_t >.
Referenced by out().

in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline]
```

Convert from external to internal character set.
Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.
The source and destination character sets are determined by the facet's locale, internal and external types.
The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.
The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.
The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

References do_out().

unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
             state_type & __state,
             extern_type * __to,
```

```
        extern_type * __to_end,
        extern_type *& __to_next) const  [inline]
```

Reset conversion state.
Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().
For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.
The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

The documentation for this class was generated from the following file:

- codecvt.h

6.10 __gnu_cxx::__common_pool_policy< _PoolTp, _Thread > Struct Template Reference

`#include <mt_allocator.h>`

6.10.1 Detailed Description

template<template< bool > class _PoolTp, bool _Thread>
struct __gnu_cxx::__common_pool_policy< _PoolTp, _Thread >

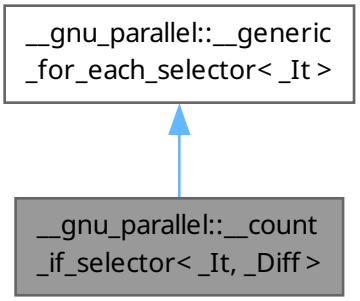
Policy for shared __pool objects.
The documentation for this struct was generated from the following file:

- mt_allocator.h

6.11 __gnu_parallel::__count_if_selector< _It, _Diff > Struct Template Reference

`#include <for_each_selectors.h>`

Inheritance diagram for __gnu_parallel::__count_if_selector< _It, _Diff >:



Public Member Functions

- template<typename _Op>
  _Diff operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

6.11.1 Detailed Description

template<typename _It, typename _Diff>
struct __gnu_parallel::__count_if_selector< _It, _Diff >

std::count_if () selector.

6.11.2 Member Function Documentation

operator()()

```
template<typename _It, typename _Diff>
template<typename _Op>
_Diff __gnu_parallel::__count_if_selector< _It, _Diff >::operator() (
            _Op & __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator.

_↩ _i	iterator referencing object.

Returns

1 if count, 0 if does not count.

6.11.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

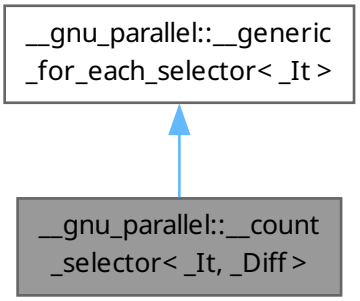
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.12 __gnu_parallel::__count_selector< _It, _Diff > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__count_selector< _It, _Diff >:



Public Member Functions

- template<typename _ValueType>
  _Diff operator() (_ValueType &__v, _It __i)

Public Attributes

- _It _M_finish_iterator

6.12.1 Detailed Description

template<typename _It, typename _Diff>
struct __gnu_parallel::__count_selector< _It, _Diff >

std::count() selector.

6.12.2 Member Function Documentation

operator()()

```
template<typename _It, typename _Diff>
template<typename _ValueType>
_Diff __gnu_parallel::__count_selector< _It, _Diff >::operator() (
            _ValueType & __v,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _v	Current value.
_↩ _i	iterator referencing object.

Returns

1 if count, 0 if does not count.

6.12.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

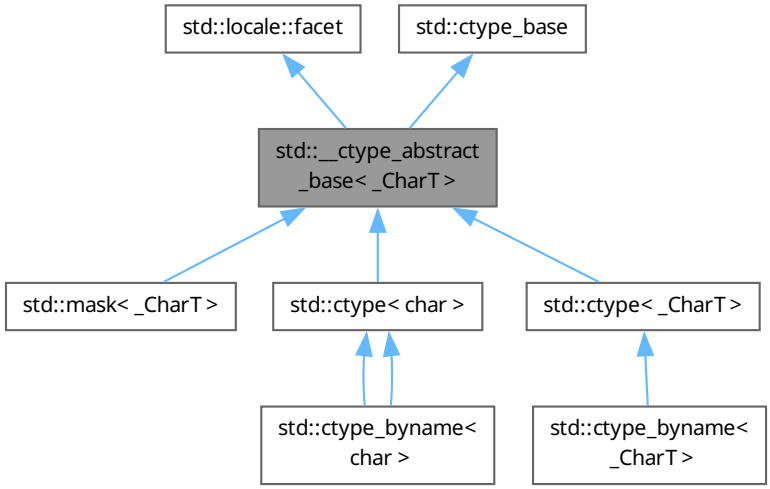
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.13 std::__ctype_abstract_base< _CharT > Class Template Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::__ctype_abstract_base< _CharT >:



Public Types

- typedef const int * **__to_type**
- typedef _CharT char_type
- typedef unsigned short **mask**

Public Member Functions

- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- char_type widen (char __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**

- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- **__ctype_abstract_base** (size_t __refs=0)
- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const =0
- virtual bool do_is (mask __m, char_type __c) const =0
- virtual char do_narrow (char_type __c, char __dfault) const =0
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const =0
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const =0
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const =0
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const =0
- virtual char_type do_tolower (char_type __c) const =0
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const =0
- virtual char_type do_toupper (char_type __c) const =0
- virtual char_type do_widen (char __c) const =0
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__to) const =0

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.13.1 Detailed Description

template<typename _CharT>
class std::__ctype_abstract_base< _CharT >

Common base for ctype facet.
This template class provides implementations of the public functions that forward to the protected virtual functions.
This template also provides abstract stubs for the protected virtual functions.

6.13.2 Member Typedef Documentation

char_type

```
template<typename _CharT>
typedef _CharT std::__ctype_abstract_base< _CharT >::char_type
```

Typedef for the template parameter.

6.13.3 Member Function Documentation

do_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [pure virtual]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implemented in std::ctype< _CharT >, and std::mask< _CharT >.

do_is() [2/2]

```
template<typename _CharT>
virtual bool std::__ctype_abstract_base< _CharT >::do_is (
            mask __m,
            char_type __c) const  [protected], [pure virtual]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The char_type to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::is(), and std::mask< _CharT >::is().

do_narrow() [1/2]

```
template<typename _CharT>
virtual char std::__ctype_abstract_base< _CharT >::do_narrow (
            char_type __c,
            char __dfault) const  [protected], [pure virtual]
```

Narrow char_type to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::narrow(), and std::mask< _CharT >::narrow().

do_narrow() [2/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [protected], [pure virtual]
```

Narrow char_type array to char.
This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implemented in std::ctype< _CharT >, and std::mask< _CharT >.

do_scan_is()

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *__hi*.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::scan_is().

do_scan_not()

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.

_↩ _hi	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __hi.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::scan_not().

do_tolower() [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual]
```

Convert array to lowercase.
This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implemented in std::ctype< _CharT >, and std::mask< _CharT >.

do_tolower() [2/2]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_tolower (
            char_type __c) const  [protected], [pure virtual]
```

Convert to lowercase.
This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::tolower(), and std::mask< _CharT >::tolower().

do_toupper() [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual]
```

Convert array to uppercase.
This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Implemented in std::ctype< _CharT >, and std::mask< _CharT >.

do_toupper() [2/2]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_toupper (
            char_type __c) const  [protected], [pure virtual]
```

Convert to uppercase.
This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

__c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::toupper(), and std::mask< _CharT >::toupper().

do_widen() [1/2]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_widen (
            char __c) const  [protected], [pure virtual]
```

Widen char.
This virtual function converts the char to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type

Implemented in std::ctype< _CharT >, std::ctype< char >, std::ctype< wchar_t >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::widen(), and std::mask< _CharT >::widen().

do_widen() [2/2]

```
template<typename _CharT>
virtual const char * std::__ctype_abstract_base< _CharT >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [protected], [pure virtual]
```

Widen char array.
This function converts each char in the input to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

Implemented in std::ctype< _CharT >, and std::mask< _CharT >.

is() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
```

```
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [2/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
            mask __m,
            char_type __c) const  [inline]
```

Test char_type classification.
This function finds a mask M for *__c* and compares it to mask *__m*. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _c	The char_type to compare the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

Referenced by std::time_get< _CharT, _InIter >::get().

narrow() [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Referenced by std::time_get< _CharT, _InIter >::do_get_year(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

narrow() [2/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

scan_is()

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else __*hi*.

scan_not()

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __*hi*.

tolower() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to lowercase.

This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
            char_type __c) const  [inline]
```

Convert to lowercase.
This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

Referenced by std::time_get< _CharT, _InIter >::get().

toupper() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
            char_type __c) const  [inline]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

Referenced by std::time_get< _CharT, _InIter >::get().

widen() [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::widen (
            char __c) const  [inline]
```

Widen char to char_type.
This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::money_put< _CharT, _Out and std::time_put< _CharT, _OutIter >::do_put().

widen() [2/2]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

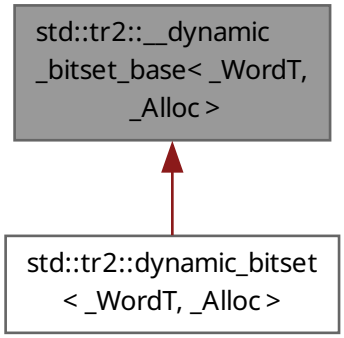
__hi.

The documentation for this class was generated from the following file:

- locale_facets.h

6.14 std::tr2::__dynamic_bitset_base< _WordT, _Alloc > Struct Template Reference

`#include <dynamic_bitset>`
Inheritance diagram for std::tr2::__dynamic_bitset_base< _WordT, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _WordT **block_type**
- typedef size_t **size_type**

Public Member Functions

- **__dynamic_bitset_base** (__dynamic_bitset_base &&__b)=default
- **__dynamic_bitset_base** (const __dynamic_bitset_base &)=default
- **__dynamic_bitset_base** (const allocator_type &__alloc)
- **__dynamic_bitset_base** (size_type __nbits, unsigned long long __val=0ULL, const allocator_type &__alloc=allocator_type())

- size_t **_M_are_all_aux** () const noexcept
- void **_M_clear** () noexcept
- void **_M_do_and** (const __dynamic_bitset_base &__x) noexcept
- void **_M_do_append_block** (block_type __block, size_type __pos)
- size_t **_M_do_count** () const noexcept
- void **_M_do_dif** (const __dynamic_bitset_base &__x) noexcept
- size_type **_M_do_find_first** (size_t __not_found) const
- size_type **_M_do_find_next** (size_t __prev, size_t __not_found) const
- void **_M_do_flip** () noexcept
- void **_M_do_left_shift** (size_t __shift)
- void **_M_do_or** (const __dynamic_bitset_base &__x) noexcept
- void **_M_do_reset** () noexcept
- void **_M_do_right_shift** (size_t __shift)
- void **_M_do_set** () noexcept
- unsigned long long **_M_do_to_ullong** () const
- unsigned long **_M_do_to_ulong** () const
- void **_M_do_xor** (const __dynamic_bitset_base &__x) noexcept
- allocator_type **_M_get_allocator** () const noexcept
- block_type **_M_getword** (size_type __pos) const noexcept
- block_type & **_M_getword** (size_type __pos) noexcept
- block_type **_M_hiword** () const noexcept
- block_type & **_M_hiword** () noexcept
- bool **_M_is_any** () const noexcept
- bool **_M_is_equal** (const __dynamic_bitset_base &__x) const noexcept
- bool **_M_is_less** (const __dynamic_bitset_base &__x) const noexcept
- bool **_M_is_proper_subset_of** (const __dynamic_bitset_base &__b) const noexcept
- bool **_M_is_subset_of** (const __dynamic_bitset_base &__b) noexcept
- void **_M_resize** (size_t __nbits, bool __value)
- size_type **_M_size** () const noexcept
- void **_M_swap** (__dynamic_bitset_base &__b) noexcept
- __dynamic_bitset_base & **operator=** (__dynamic_bitset_base &&)=default
- __dynamic_bitset_base & **operator=** (const __dynamic_bitset_base &)=default

Static Public Member Functions

- static block_type **_S_maskbit** (size_type __pos) noexcept
- static size_type **_S_whichbit** (size_type __pos) noexcept
- static size_type **_S_whichbyte** (size_type __pos) noexcept
- static size_type **_S_whichword** (size_type __pos) noexcept

Public Attributes

- std::vector< block_type, allocator_type > _M_w

Static Public Attributes

- static const size_type **_S_bits_per_block**
- static const size_type **npos**

6.14.1 Detailed Description

template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
struct std::tr2::__dynamic_bitset_base< _WordT, _Alloc >

Base class, general case.
See documentation for dynamic_bitset.

6.14.2 Member Data Documentation

_M_w

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::vector<block_type, allocator_type> std::tr2::__dynamic_bitset_base< _WordT, _Alloc >::_M_w
```

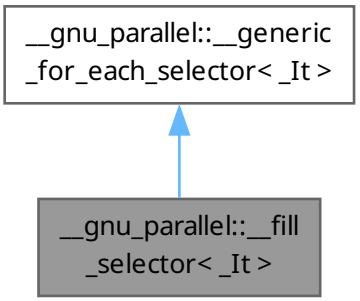
0 is the least significant word.
The documentation for this struct was generated from the following files:

- dynamic_bitset
- dynamic_bitset.tcc

6.15 __gnu_parallel::__fill_selector< _It > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__fill_selector< _It >:



Public Member Functions

- template<typename _ValueType>
  bool operator() (_ValueType &__v, _It __i)

Public Attributes

- _It _M_finish_iterator

6.15.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__fill_selector< _It >

std::fill() selector.

6.15.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _ValueType>
bool __gnu_parallel::__fill_selector< _It >::operator() (
            _ValueType & __v,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _v	Current value.
_↩ _i	iterator referencing object.

6.15.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

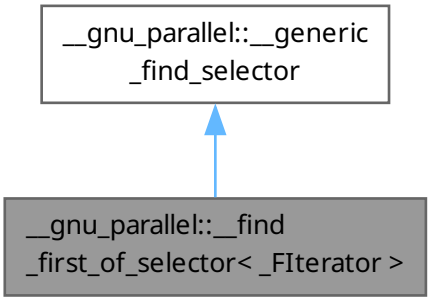
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.16 __gnu_parallel::__find_first_of_selector< _FIterator > Struct Template Reference

```
#include <find_selectors.h>
```

Inheritance diagram for __gnu_parallel::__find_first_of_selector< _FIterator >:



Public Member Functions

- **__find_first_of_selector** (_FIterator __begin, _FIterator __end)
- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __↩
  begin2, _Pred __pred)
- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)

Public Attributes

- _FIterator **_M_begin**
- _FIterator **_M_end**

6.16.1 Detailed Description

template<typename _FIterator>
struct __gnu_parallel::__find_first_of_selector< _FIterator >

Test predicate on several elements.

6.16.2 Member Function Documentation

_M_sequential_algorithm()

```
template<typename _FIterator>
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_first_of_selector< _FIterator >::_M_↩
sequential_algorithm (
            _RAIter1 __begin1,
            _RAIter1 __end1,
            _RAIter2 __begin2,
            _Pred __pred)  [inline]
```

Corresponding sequential algorithm on a sequence.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__pred	Find predicate.

References std::make_pair().

operator()()

```
template<typename _FIterator>
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__find_first_of_selector< _FIterator >::operator() (
            _RAIter1 __i1,
            _RAIter2 __i2,
            _Pred __pred)  [inline]
```

Test on one position.

Parameters

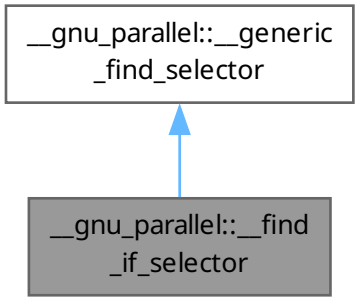
__i1	_Iterator on first sequence.
__i2	_Iterator on second sequence (unused).
__pred	Find predicate.

The documentation for this struct was generated from the following file:

- find_selectors.h

6.17 __gnu_parallel::__find_if_selector Struct Reference

`#include <find_selectors.h>`
Inheritance diagram for __gnu_parallel::__find_if_selector:



Public Member Functions

- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __↩
  begin2, _Pred __pred)
- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)

6.17.1 Detailed Description

Test predicate on a single element, used for std::find() and std::find_if ().

6.17.2 Member Function Documentation

_M_sequential_algorithm()

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_if_selector::_M_sequential_algorithm (
            _RAIter1 __begin1,
            _RAIter1 __end1,
```

```
            _RAIter2 __begin2,
            _Pred __pred)  [inline]
```

Corresponding sequential algorithm on a sequence.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__pred	Find predicate.

References std::make_pair().

operator()()

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__find_if_selector::operator() (
            _RAIter1 __i1,
            _RAIter2 __i2,
            _Pred __pred)  [inline]
```

Test on one position.

Parameters

__i1	_Iterator on first sequence.
__i2	_Iterator on second sequence (unused).
__pred	Find predicate.

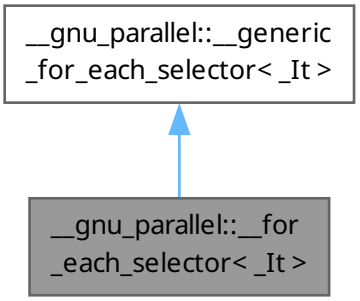
The documentation for this struct was generated from the following file:

- find_selectors.h

6.18 __gnu_parallel::__for_each_selector< _It > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__for_each_selector< _It >:



Public Member Functions

- template<typename _Op>
  bool operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

6.18.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__for_each_selector< _It >

std::for_each() selector.

6.18.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__for_each_selector< _It >::operator() (
            _Op & __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator.

_↩ _i	iterator referencing object.

6.18.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.19 __cxxabiv1::__forced_unwind Class Reference

```
#include <cxxabi_forced.h>
```

6.19.1 Detailed Description

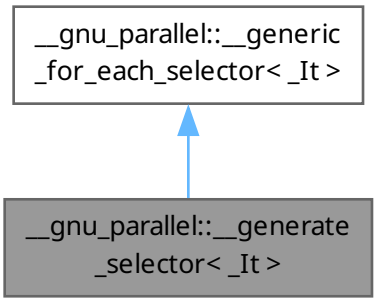
Thrown as part of forced unwinding.
A magic placeholder class that can be caught by reference to recognize forced unwinding.
The documentation for this class was generated from the following file:

- cxxabi_forced.h

6.20 __gnu_parallel::__generate_selector< _It > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__generate_selector< _It >:



Public Member Functions

- template<typename _Op>
  bool operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

6.20.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__generate_selector< _It >

std::generate() selector.

6.20.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__generate_selector< _It >::operator() (
            _Op & __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator.
_↩ _i	iterator referencing object.

6.20.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

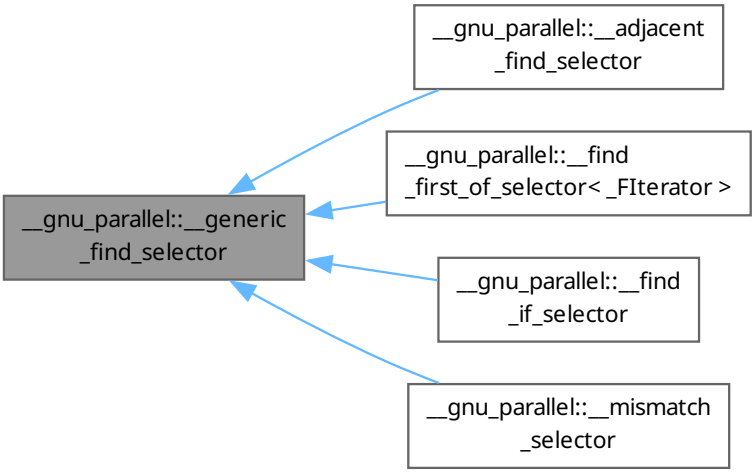
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.21 __gnu_parallel::__generic_find_selector Struct Reference

```
#include <find_selectors.h>
```

Inheritance diagram for __gnu_parallel::__generic_find_selector:



6.21.1 Detailed Description

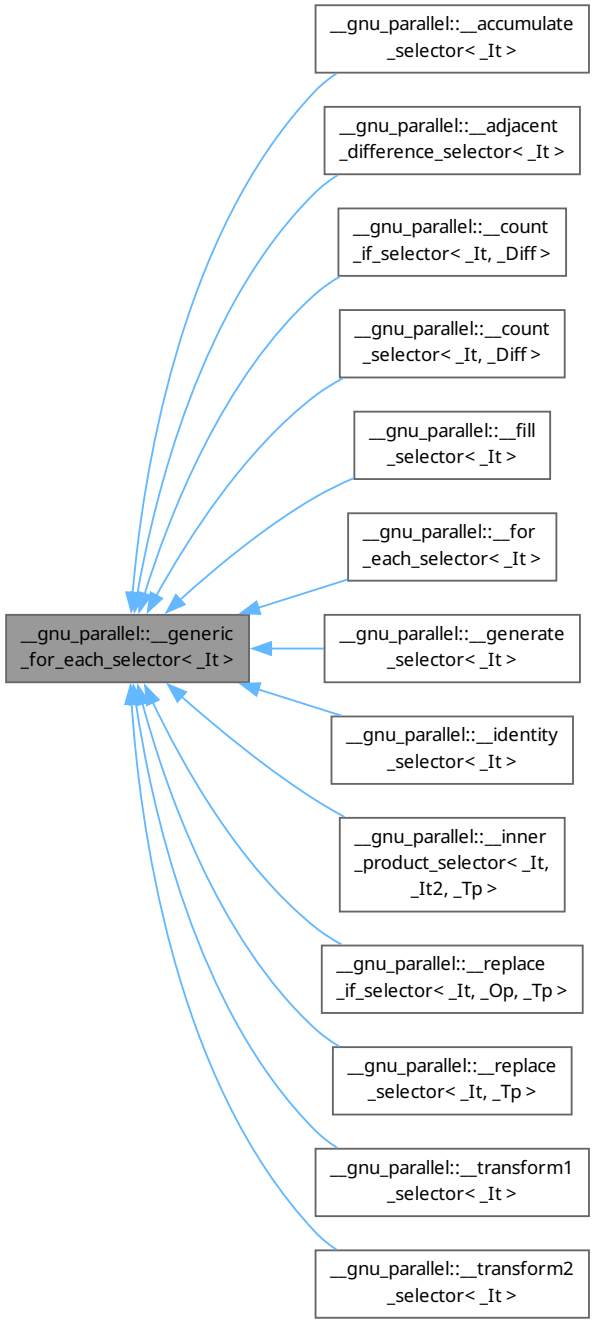
Base class of all __gnu_parallel::__find_template selectors.
The documentation for this struct was generated from the following file:

- find_selectors.h

6.22 __gnu_parallel::__generic_for_each_selector< _It > Struct Template Reference

`#include <for_each_selectors.h>`

Inheritance diagram for __gnu_parallel::__generic_for_each_selector< _It >:



Public Attributes

- _It _M_finish_iterator

6.22.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__generic_for_each_selector< _It >

Generic __selector for embarrassingly parallel functions.

6.22.2 Member Data Documentation

_M_finish_iterator

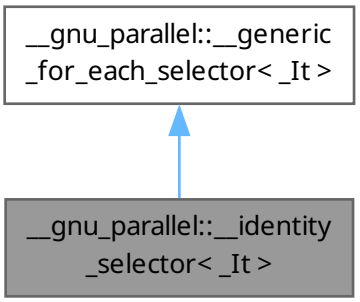
```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator
```

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.23 __gnu_parallel::__identity_selector< _It > Struct Template Reference

`#include <for_each_selectors.h>`
Inheritance diagram for __gnu_parallel::__identity_selector< _It >:



Public Member Functions

- template<typename _Op>
  _It operator() (_Op __o, _It __i)

Public Attributes

- _It _M_finish_iterator

6.23.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__identity_selector< _It >

Selector that just returns the passed iterator.

6.23.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _Op>
_It __gnu_parallel::__identity_selector< _It >::operator() (
            _Op __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator (unused).
_↩ _i	iterator referencing object.

Returns

Passed iterator.

6.23.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

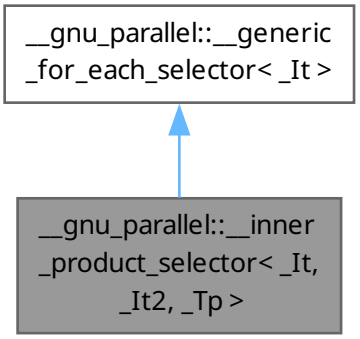
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.24 __gnu_parallel::__inner_product_selector< _It, _It2, _Tp > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >:



Public Member Functions

- __inner_product_selector (_It __b1, _It2 __b2)
- template<typename _Op>
  _Tp operator() (_Op __mult, _It __current)

Public Attributes

- _It __begin1_iterator
- _It2 __begin2_iterator
- _It _M_finish_iterator

6.24.1 Detailed Description

template<typename _It, typename _It2, typename _Tp>
struct __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >

std::inner_product() selector.

6.24.2 Constructor & Destructor Documentation

__inner_product_selector()

```
template<typename _It, typename _It2, typename _Tp>
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__inner_product_selector (
            _It __b1,
            _It2 __b2)  [inline], [explicit]
```

Constructor.

Parameters

__b1	Begin iterator of first sequence.
__b2	Begin iterator of second sequence.

References __begin1_iterator, and __begin2_iterator.

6.24.3 Member Function Documentation

operator()()

```
template<typename _It, typename _It2, typename _Tp>
template<typename _Op>
_Tp __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::operator() (
            _Op __mult,
            _It __current)  [inline]
```

Functor execution.

Parameters

__mult	Multiplication functor.
__current	iterator referencing object.

Returns

Inner product elemental __result.

References __begin1_iterator, and __begin2_iterator.

6.24.4 Member Data Documentation

__begin1_iterator

```
template<typename _It, typename _It2, typename _Tp>
_It __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin1_iterator
```

Begin iterator of first sequence.
Referenced by __inner_product_selector(), and operator()().

__begin2_iterator

```
template<typename _It, typename _It2, typename _Tp>
_It2 __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >::__begin2_iterator
```

Begin iterator of second sequence.
Referenced by __inner_product_selector(), and operator()().

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.25 std::__is_fast_hash< _Hash > Struct Template Reference

```
#include <functional_hash.h>
```

6.25.1 Detailed Description

template<typename _Hash>
struct std::__is_fast_hash< _Hash >

Hint about performance of hash functions.
If a given hash function object is not fast, the hash-based containers will cache the hash code. The default behavior is to consider that hashers are fast unless specified otherwise.
Users can specialize this for their own hash functions in order to force caching of hash codes in unordered containers. Specializing this trait affects the ABI of the unordered containers, so use it carefully.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.26 std::__is_location_invariant< _Tp > Struct Template Reference

`#include <std_function.h>`

6.26.1 Detailed Description

template<typename _Tp>
struct std::__is_location_invariant< _Tp >

Trait identifying "location-invariant" types, meaning that the address of the object (or any of its members) will not escape. Trivially copyable types are location-invariant and users can specialize this trait for other types.
The documentation for this struct was generated from the following file:

- std_function.h

6.27 __gnu_parallel::__max_element_reduct< _Compare, _It > Struct Template Reference

`#include <for_each_selectors.h>`

Public Member Functions

- **__max_element_reduct** (_Compare &__c)
- _It **operator()** (_It __x, _It __y)

Public Attributes

- _Compare & **__comp**

6.27.1 Detailed Description

template<typename _Compare, typename _It>
struct __gnu_parallel::__max_element_reduct< _Compare, _It >

Reduction for finding the maximum element, using a comparator.
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.28 __gnu_parallel::__min_element_reduct< _Compare, _It > Struct Template Reference

`#include <for_each_selectors.h>`

Public Member Functions

- **__min_element_reduct** (_Compare &__c)
- _It **operator()** (_It __x, _It __y)

Public Attributes

- _Compare & **__comp**

6.28.1 Detailed Description

template<typename _Compare, typename _It>
struct __gnu_parallel::__min_element_reduct< _Compare, _It >

Reduction for finding the maximum element, using a comparator.
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.29 __gnu_cxx::__detail::__mini_vector< _Tp > Class Template Reference

`#include <bitmap_allocator.h>`

Public Types

- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef pointer **iterator**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- reference **back** () const throw ()
- iterator **begin** () const throw ()
- void **clear** () throw ()
- iterator **end** () const throw ()
- void **erase** (iterator __pos) throw ()
- void **insert** (iterator __pos, const_reference __x)
- reference **operator[ ]** (const size_type __pos) const throw ()
- void **pop_back** () throw ()
- void **push_back** (const_reference __x)
- size_type **size** () const throw ()

6.29.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::__detail::__mini_vector< _Tp >

__mini_vector<> is a stripped down version of the full-fledged std::vector<>.
It is to be used only for built-in types or PODs. Notable differences are:

1. Not all accessor functions are present.

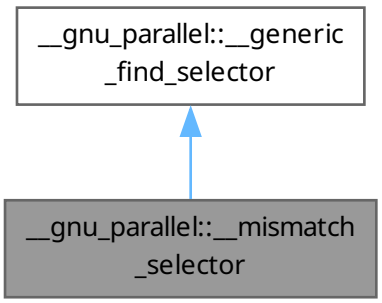
2. Used ONLY for PODs.
3. No Allocator template argument. Uses operator new() to get memory, and operator delete() to free it. Caveat: The dtor does NOT free the memory allocated, so this a memory-leaking vector!

The documentation for this class was generated from the following file:

- bitmap_allocator.h

6.30 __gnu_parallel::__mismatch_selector Struct Reference

`#include <find_selectors.h>`
Inheritance diagram for __gnu_parallel::__mismatch_selector:



Public Member Functions

- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  std::pair< _RAIter1, _RAIter2 > _M_sequential_algorithm (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred)
- template<typename _RAIter1, typename _RAIter2, typename _Pred>
  bool operator() (_RAIter1 __i1, _RAIter2 __i2, _Pred __pred)

6.30.1 Detailed Description

Test inverted predicate on a single element.

6.30.2 Member Function Documentation

_M_sequential_algorithm()

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__mismatch_selector::_M_sequential_algorithm (
            _RAIter1 __begin1,
            _RAIter1 __end1,
            _RAIter2 __begin2,
            _Pred __pred)  [inline]
```

Corresponding sequential algorithm on a sequence.

Parameters

__begin1	Begin iterator of first sequence.
__end1	End iterator of first sequence.
__begin2	Begin iterator of second sequence.
__pred	Find predicate.

operator()()

```
template<typename _RAIter1, typename _RAIter2, typename _Pred>
bool __gnu_parallel::__mismatch_selector::operator() (
             _RAIter1 __i1,
             _RAIter2 __i2,
             _Pred __pred)  [inline]
```

Test on one position.

Parameters

__i1	_Iterator on first sequence.
__i2	_Iterator on second sequence (unused).
__pred	Find predicate.

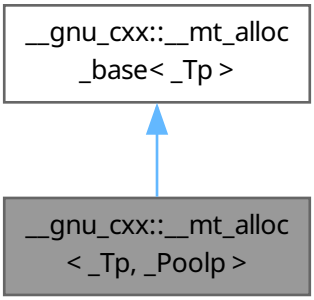
The documentation for this struct was generated from the following file:

- find_selectors.h

6.31 __gnu_cxx::__mt_alloc< _Tp, _Poolp > Class Template Reference

`#include <mt_allocator.h>`

Inheritance diagram for __gnu_cxx::__mt_alloc< _Tp, _Poolp >:



Public Types

- typedef _Poolp **__policy_type**
- typedef _Poolp::pool_type **__pool_type**
- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **__mt_alloc** (const __mt_alloc &) noexcept
- template<typename _Tp1, typename _Poolp1>
  __mt_alloc (const __mt_alloc< _Tp1, _Poolp1 > &) noexcept
- const __pool_base::_Tune **_M_get_options** ()
- void **_M_set_options** (__pool_base::_Tune __t)
- const_pointer **address** (const_reference __x) const noexcept
- pointer **address** (reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *=0)
- template<typename _Up, typename... _Args>
  void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up>
  void **destroy** (_Up *__p)
- size_type **max_size** () const noexcept

6.31.1 Detailed Description

template<typename _Tp, typename _Poolp = __common_pool_policy<__pool, true >>
class __gnu_cxx::__mt_alloc< _Tp, _Poolp >

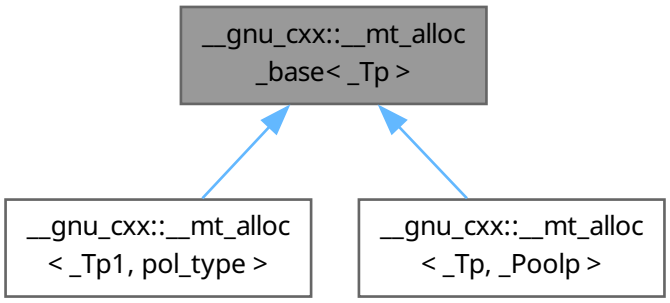
This is a fixed size (power of 2) allocator which - when compiled with thread support - will maintain one freelist per size per thread plus a *global* one. Steps are taken to limit the per thread freelist sizes (by returning excess back to the *global* list).
Further details: https://gcc.gnu.org/onlinedocs/libstdc++/manual/mt_allocator.html
The documentation for this class was generated from the following file:

- mt_allocator.h

6.32 __gnu_cxx::__mt_alloc_base< _Tp > Class Template Reference

`#include <mt_allocator.h>`

Inheritance diagram for __gnu_cxx::__mt_alloc_base< _Tp >:



Public Types

- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- const_pointer **address** (const_reference __x) const noexcept
- pointer **address** (reference __x) const noexcept
- template<typename _Up, typename... _Args>
  void **construct** (_Up *__p, _Args &&... __args)
- template<typename _Up>
  void **destroy** (_Up *__p)
- size_type **max_size** () const noexcept

6.32.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::__mt_alloc_base< _Tp >

Base class for _Tp dependent member functions.
The documentation for this class was generated from the following file:

- mt_allocator.h

6.33 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAlterIterator, _RAlter3, _DifferenceTp, _Compare > Struct Template Reference

```
#include <multiway_merge.h>
```

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _↩ DifferenceTp __length, _Compare __comp)

6.33.1 Detailed Description

template<bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

Switch for 3-way merging with __sentinels turned off.
Note that 3-way merging is always stable!
The documentation for this struct was generated from the following file:

- multiway_merge.h

6.34 __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _↩ DifferenceTp __length, _Compare __comp)

6.34.1 Detailed Description

template<typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _↩ DifferenceTp, _Compare >

Switch for 3-way merging with __sentinels turned on.
Note that 3-way merging is always stable!
The documentation for this struct was generated from the following file:

- multiway_merge.h

6.35 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _↩ DifferenceTp __length, _Compare __comp)

6.35.1 Detailed Description

template<bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

Switch for 4-way merging with __sentinels turned off.

Note that 4-way merging is always stable!
The documentation for this struct was generated from the following file:

- multiway_merge.h

6.36 __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAlterIterator, _RAlter3, _DifferenceTp, _Compare > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- _RAlter3 **operator()** (_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _RAlter3 __target, _DifferenceTp __length, _Compare __comp)

6.36.1 Detailed Description

template<typename _RAlterIterator, typename _RAlter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAlterIterator, _RAlter3, _DifferenceTp, _Compare >

Switch for 4-way merging with __sentinels turned on.
Note that 4-way merging is always stable!
The documentation for this struct was generated from the following file:

- multiway_merge.h

6.37 __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAlterIterator, _RAlter3, _DifferenceTp, _Compare > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- _RAlter3 **operator()** (_RAlterIterator __seqs_begin, _RAlterIterator __seqs_end, _RAlter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAlterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)

6.37.1 Detailed Description

template<bool __sentinels, bool __stable, typename _RAlterIterator, typename _RAlter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAlterIterator, _RAlter3, _DifferenceTp, _Compare >

Switch for k-way merging with __sentinels turned on.
The documentation for this struct was generated from the following file:

- multiway_merge.h

6.38 __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAlterIterator, _RAlter3, _DifferenceTp, _Compare > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- _RAIter3 **operator()** (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)

6.38.1 Detailed Description

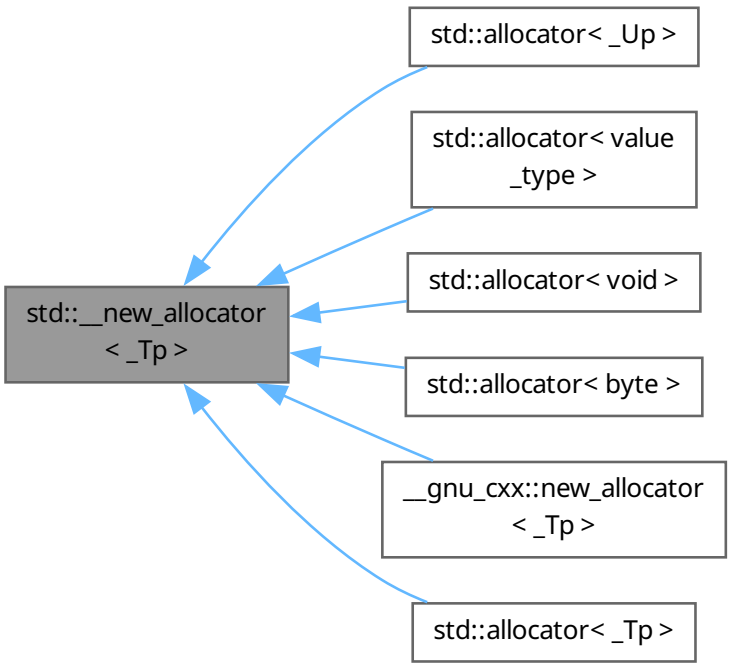
template<bool __stable, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >

Switch for k-way merging with __sentinels turned off.
The documentation for this struct was generated from the following file:

- multiway_merge.h

6.39 std::__new_allocator< _Tp > Class Template Reference

`#include <memory>`
Inheritance diagram for std::__new_allocator< _Tp >:



Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::true_type **propagate_on_container_move_assignment**

- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **__new_allocator** (const __new_allocator &) noexcept
- template<typename _Tp1>
  constexpr **__new_allocator** (const __new_allocator< _Tp1 > &) noexcept
- constexpr _Tp * **allocate** (size_type __n, const void *=static_cast< const void * >(0))
- constexpr std::allocation_result< _Tp *, size_t > **allocate_at_least** (size_t __n)
- constexpr void **deallocate** (_Tp *__p, size_type __n)
- __new_allocator & **operator=** (const __new_allocator &)=default

Static Public Member Functions

- static constexpr void **_S_check_allocation_limit** (size_t __n)

Friends

- template<typename _Up>
  constexpr bool **operator==** (const __new_allocator &, const __new_allocator< _Up > &) noexcept

6.39.1 Detailed Description

template<typename _Tp>
class std::__new_allocator< _Tp >

An allocator that uses global `new`, as per C++03 [20.4.1].
This is precisely the allocator defined in the C++ Standard.

- all allocation calls `operator new`
- all deallocation calls `operator delete`

This is the default base-class implementation of `std::allocator`, and is also the base-class of the `__gnu_cxx::new_allocator` extension. You should use either `std::allocator` or `__gnu_cxx::new_allocator` instead of using this directly.

Template Parameters

_Tp	Type of allocated object.

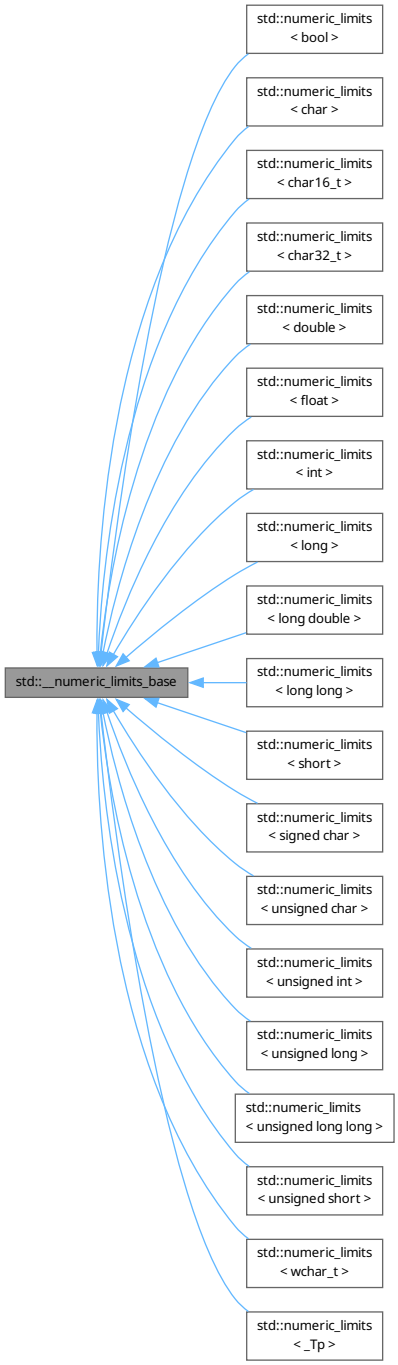
The documentation for this class was generated from the following file:

- bits/new_allocator.h

6.40 std::__numeric_limits_base Struct Reference

`#include <limits>`

Inheritance diagram for std::__numeric_limits_base:



Static Public Attributes

- static constexpr int digits
- static constexpr int digits10

- static constexpr float_denorm_style has_denorm
- static constexpr bool has_denorm_loss
- static constexpr bool has_infinity
- static constexpr bool has_quiet_NaN
- static constexpr bool has_signaling_NaN
- static constexpr bool is_bounded
- static constexpr bool is_exact
- static constexpr bool is_iec559
- static constexpr bool is_integer
- static constexpr bool is_modulo
- static constexpr bool is_signed
- static constexpr bool is_specialized
- static constexpr int max_digits10
- static constexpr int max_exponent
- static constexpr int max_exponent10
- static constexpr int min_exponent
- static constexpr int min_exponent10
- static constexpr int radix
- static constexpr float_round_style round_style
- static constexpr bool tinyness_before
- static constexpr bool traps

6.40.1 Detailed Description

Part of std::numeric_limits.
The `static const` members are usable as integral constant expressions.

Note

This is a separate class for purposes of efficiency; you should only access these members as part of an instantiation of the std::numeric_limits class.

6.40.2 Member Data Documentation

digits

`int std::__numeric_limits_base::digits [static], [constexpr]`
The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.
Referenced by std::generate_canonical(), std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator()(), and std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed().

digits10

`int std::__numeric_limits_base::digits10 [static], [constexpr]`
The number of base 10 digits that can be represented without change.

has_denorm

`float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]`
See std::float_denorm_style for more information.

has_denorm_loss

`bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]`
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

`bool std::__numeric_limits_base::has_infinity [static], [constexpr]`
True if the type has a representation for positive infinity.

has_quiet_NaN

`bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]`
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

`bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]`
True if the type has a representation for a signaling Not a Number.

is_bounded

`bool std::__numeric_limits_base::is_bounded [static], [constexpr]`
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

`bool std::__numeric_limits_base::is_exact [static], [constexpr]`
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

`bool std::__numeric_limits_base::is_iec559 [static], [constexpr]`
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.
Referenced by std::generate_canonical().

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

traps

`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.41 __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread > Struct Template Reference

`#include <mt_allocator.h>`

6.41.1 Detailed Description

template<typename _Tp, template< bool > class _PoolTp, bool _Thread>
struct __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >

Policy for individual __pool objects.
The documentation for this struct was generated from the following file:

- mt_allocator.h

6.42 __gnu_cxx::__pool< _Thread > Class Template Reference

6.42.1 Detailed Description

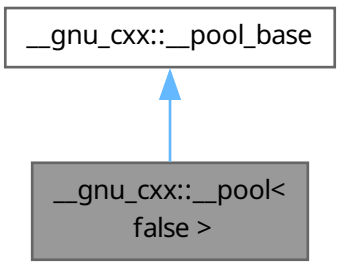
template<bool _Thread>
class __gnu_cxx::__pool< _Thread >

Data describing the underlying memory pool, parameterized on threading support.
The documentation for this class was generated from the following file:

- mt_allocator.h

6.43 __gnu_cxx::__pool< false > Class Reference

`#include <mt_allocator.h>`
Inheritance diagram for __gnu_cxx::__pool< false >:



Public Types

- typedef unsigned short int **_Binmap_type**
- typedef std::size_t **size_t**

Public Member Functions

- **__pool** (const __pool_base::_Tune &__tune)
- void **_M_adjust_freelist** (const _Bin_record &, _Block_record *, size_t)
- bool **_M_check_threshold** (size_t __bytes)
- void **_M_destroy** () throw ()
- size_t **_M_get_align** ()
- const _Bin_record & **_M_get_bin** (size_t __which)
- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- size_t **_M_get_thread_id** ()
- void **_M_initialize_once** ()
- void **_M_reclaim_block** (char *__p, size_t __bytes) throw ()
- char * **_M_reserve_block** (size_t __bytes, const size_t __thread_id)
- void **_M_set_options** (_Tune __t)

Protected Attributes

- _Binmap_type * **_M_binmap**
- bool **_M_init**
- _Tune **_M_options**

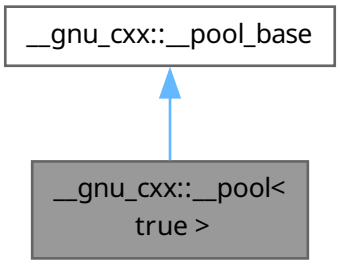
6.43.1 Detailed Description

Specialization for single thread.
The documentation for this class was generated from the following file:

- mt_allocator.h

6.44 __gnu_cxx::__pool< true > Class Reference

`#include <mt_allocator.h>`
Inheritance diagram for __gnu_cxx::__pool< true >:



Public Types

- typedef unsigned short int **_Binmap_type**
- typedef std::size_t **size_t**

Public Member Functions

- **__pool** (const __pool_base::_Tune &__tune)
- void **_M_adjust_freelist** (const _Bin_record &__bin, _Block_record *__block, size_t __thread_id)
- bool **_M_check_threshold** (size_t __bytes)
- void **_M_destroy** () throw ()
- void **_M_destroy_thread_key** (void *) throw ()
- size_t **_M_get_align** ()
- const _Bin_record & **_M_get_bin** (size_t __which)
- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- size_t **_M_get_thread_id** ()
- void **_M_initialize** (__destroy_handler)
- void **_M_initialize_once** ()
- void **_M_reclaim_block** (char *__p, size_t __bytes) throw ()
- char * **_M_reserve_block** (size_t __bytes, const size_t __thread_id)
- void **_M_set_options** (_Tune __t)

Protected Attributes

- _Binmap_type * **_M_binmap**
- bool **_M_init**
- _Tune **_M_options**

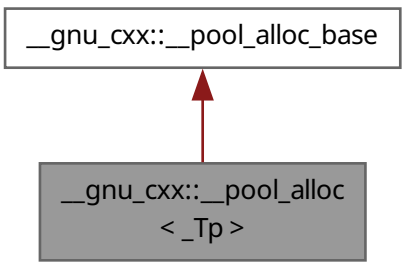
6.44.1 Detailed Description

Specialization for thread enabled, via gthreads.h.
The documentation for this class was generated from the following file:

- mt_allocator.h

6.45 __gnu_cxx::__pool_alloc< _Tp > Class Template Reference

`#include <pool_allocator.h>`
Inheritance diagram for __gnu_cxx::__pool_alloc< _Tp >:



Public Types

- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **__pool_alloc** (const __pool_alloc &) noexcept
- template<typename _Tp1>
  __pool_alloc (const __pool_alloc< _Tp1 > &) noexcept
- const_pointer **address** (const_reference __x) const noexcept
- pointer **address** (reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *=0)
- template<typename _Up, typename... _Args>
  void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up>
  void **destroy** (_Up *__p)
- size_type **max_size** () const noexcept

6.45.1 Detailed Description

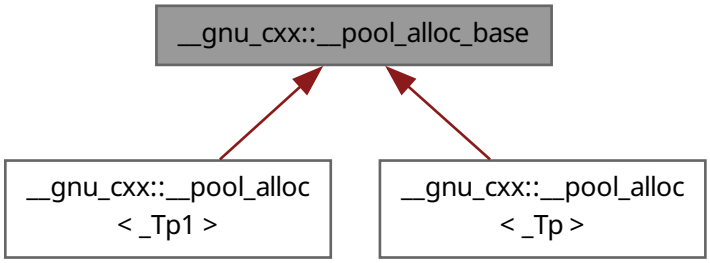
template<typename _Tp>
class __gnu_cxx::__pool_alloc< _Tp >

Allocator using a memory pool with a single lock.
The documentation for this class was generated from the following file:

- pool_allocator.h

6.46 __gnu_cxx::__pool_alloc_base Class Reference

`#include <pool_allocator.h>`
Inheritance diagram for __gnu_cxx::__pool_alloc_base:



Protected Types

- enum { **_S_free_list_size** }
- enum { **_S_max_bytes** }
- enum { **_S_align** }

Protected Member Functions

- char * **_M_allocate_chunk** (size_t __n, int &__nobjs)
- _Obj *volatile * **_M_get_free_list** (size_t __bytes) throw ()
- __mutex & **_M_get_mutex** () throw ()
- void * **_M_refill** (size_t __n)
- size_t **_M_round_up** (size_t __bytes)

Static Protected Attributes

- static char * **_S_end_free**
- static _Obj *volatile **_S_free_list** [_S_free_list_size]
- static size_t **_S_heap_size**
- static char * **_S_start_free**

6.46.1 Detailed Description

Base class for __pool_alloc.
Uses various allocators to fulfill underlying requests (and makes as few requests as possible when in default high-speed pool mode).
Important implementation properties:

1. If globally mandated, then allocate objects from new
2. If the clients request an object of size > _S_max_bytes, the resulting object will be obtained directly from new
3. In all other cases, we allocate an object of size exactly _S_round_up(requested_size). Thus the client has enough size information that we can return the object to the proper free list without permanently losing part of the object.

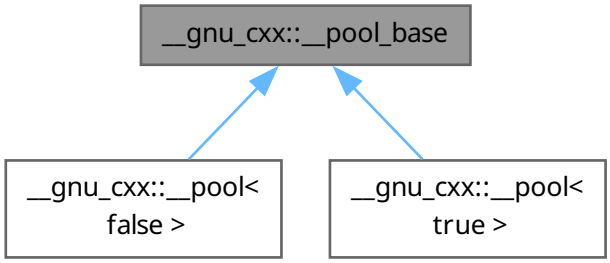
The documentation for this class was generated from the following file:

- pool_allocator.h

6.47 __gnu_cxx::__pool_base Struct Reference

`#include <mt_allocator.h>`

Inheritance diagram for __gnu_cxx::__pool_base:



Public Types

- typedef unsigned short int **_Binmap_type**
- typedef std::size_t **size_t**

Public Member Functions

- **__pool_base** (const _Tune &__options)
- bool **_M_check_threshold** (size_t __bytes)
- size_t **_M_get_align** ()
- size_t **_M_get_binmap** (size_t __bytes)
- const _Tune & **_M_get_options** () const
- void **_M_set_options** (_Tune __t)

Protected Attributes

- _Binmap_type * **_M_binmap**
- bool **_M_init**
- _Tune **_M_options**

6.47.1 Detailed Description

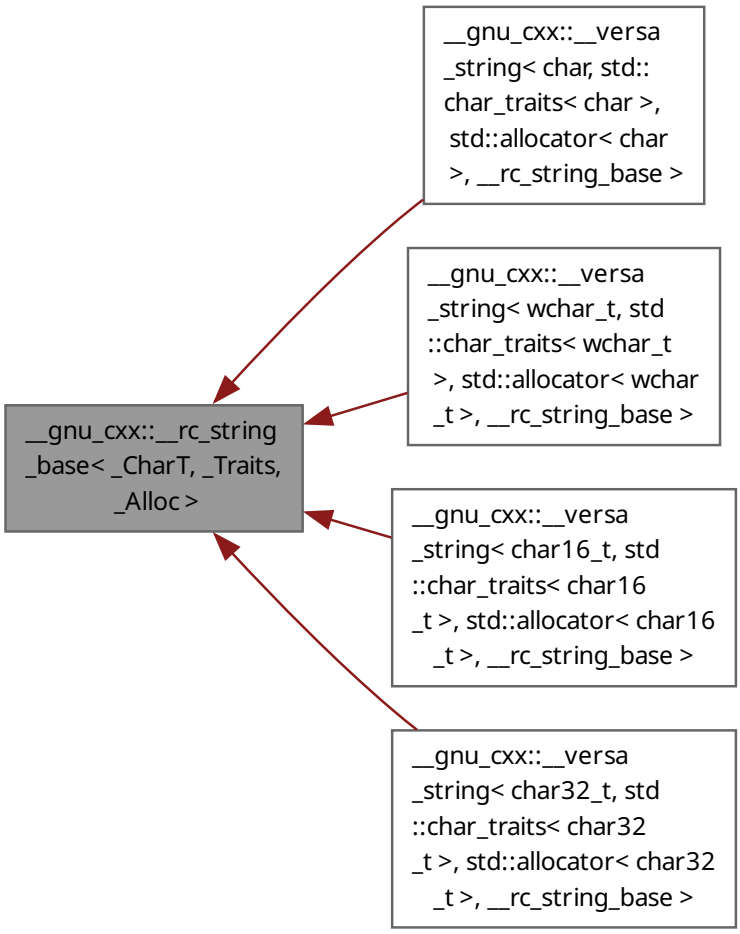
Base class for pool object.
The documentation for this struct was generated from the following file:

- mt_allocator.h

6.48 __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc > Class Template Reference

`#include <rc_string_base.h>`

Inheritance diagram for __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >:



Public Types

- typedef _Util_Base::_CharT_alloc_type **_CharT_alloc_type**
- typedef __vstring_utility< _CharT, _Traits, _Alloc > **_Util_Base**
- typedef _Alloc **allocator_type**
- typedef _CharT_alloc_type::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- **__rc_string_base** (__rc_string_base &&__rcs)
- template<typename _InputIterator>
  __rc_string_base (_InputIterator __beg, _InputIterator __end, const _Alloc &__a)
- **__rc_string_base** (const __rc_string_base &__rcs)

- **__rc_string_base** (const _Alloc &__a)
- **__rc_string_base** (size_type __n, _CharT __c, const _Alloc &__a)
- void **_M_assign** (const __rc_string_base &__rcs)
- size_type **_M_capacity** () const
- void **_M_clear** ()
- bool **_M_compare** (const __rc_string_base &) const
- bool **_M_compare** (const __rc_string_base &__rcs) const
- bool **_M_compare** (const __rc_string_base &__rcs) const
- _CharT * **_M_data** () const
- void **_M_erase** (size_type __pos, size_type __n)
- allocator_type & **_M_get_allocator** ()
- const allocator_type & **_M_get_allocator** () const
- bool **_M_is_shared** () const
- void **_M_leak** ()
- size_type **_M_length** () const
- size_type **_M_max_size** () const
- void **_M_mutate** (size_type __pos, size_type __len1, const _CharT *__s, size_type __len2)
- void **_M_reserve** (size_type __res)
- void **_M_set_leaked** ()
- void **_M_set_length** (size_type __n)
- void **_M_swap** (__rc_string_base &__rcs)
- template<typename _InIterator>
  _CharT * **_S_construct** (_InIterator __beg, _InIterator __end, const _Alloc &__a, std::forward_iterator_tag)

Protected Types

- typedef __gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _↩ _rc_string_base > > **__const_rc_iterator**
- typedef __gnu_cxx::__normal_iterator< const_pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _↩ _sso_string_base > > **__const_sso_iterator**
- typedef __gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __rc_↩ string_base > > **__rc_iterator**
- typedef __gnu_cxx::__normal_iterator< pointer, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, __sso_↩ string_base > > **__sso_iterator**
- typedef __alloc_traits< _CharT_alloc_type > **_CharT_alloc_traits**
- typedef _CharT_alloc_traits::const_pointer **const_pointer**
- typedef _CharT_alloc_type::difference_type **difference_type**
- typedef _CharT_alloc_traits::pointer **pointer**

Static Protected Member Functions

- static void **_S_assign** (_CharT *__d, size_type __n, _CharT __c)
- static int **_S_compare** (size_type __n1, size_type __n2)
- static void **_S_copy** (_CharT *__d, const _CharT *__s, size_type __n)
- static void **_S_copy_chars** (_CharT *__p, __const_rc_iterator __k1, __const_rc_iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, __const_sso_iterator __k1, __const_sso_iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, __rc_iterator __k1, __rc_iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, __sso_iterator __k1, __sso_iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, _CharT *__k1, _CharT *__k2)
- template<typename _Iterator>
  static void **_S_copy_chars** (_CharT *__p, _Iterator __k1, _Iterator __k2)
- static void **_S_copy_chars** (_CharT *__p, const _CharT *__k1, const _CharT *__k2)
- static void **_S_move** (_CharT *__d, const _CharT *__s, size_type __n)

6.48.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>
class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >

Documentation? What's that? Nathan Myers ncm@cantrip.org.
A string looks like this:

```
                                    [_Rep]
                                    _M_length
[__rc_string_base<char_type>]       _M_capacity
_M_dataplus                         _M_refcount
_M_p ---------------->              unnamed array of char_type
```

Where the _M_p points to the first character in the string, and you cast it to a pointer-to-_Rep and subtract 1 to get a pointer to the header.
This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: _Rep::_M_refdata(), and __rc_↩ string_base::_M_rep(); and the allocation function which gets a block of raw bytes and with room enough and constructs a _Rep object at the front.
The reason you want _M_data pointing to the character array and not the _Rep is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the _Rep for the debugger to use, so users can check the actual string length.)
Note that the _Rep object is a POD so that you can have a static *empty string* _Rep object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string _Rep object.
All but the last paragraph is considered pretty conventional for a C++ string implementation.
The documentation for this class was generated from the following file:

- rc_string_base.h

6.49 std::tr2::__reflection_typelist< _Elements > Struct Template Reference

6.49.1 Detailed Description

template<typename... _Elements>
struct std::tr2::__reflection_typelist< _Elements >

See N2965: Type traits and base classes by Michael Spertus Simple typelist. Compile-time list of types.
The documentation for this struct was generated from the following file:

- tr2/type_traits

6.50 std::tr2::__reflection_typelist< _First, _Rest... > Struct Template Reference

`#include <type_traits>`

Public Types

- typedef std::false_type **empty**

6.50.1 Detailed Description

template<typename _First, typename... _Rest>
struct std::tr2::__reflection_typelist< _First, _Rest... >

Partial specialization.
The documentation for this struct was generated from the following file:

- tr2/type_traits

6.51 std::tr2::__reflection_typelist<> Struct Reference

`#include <type_traits>`

Public Types

- typedef std::true_type **empty**

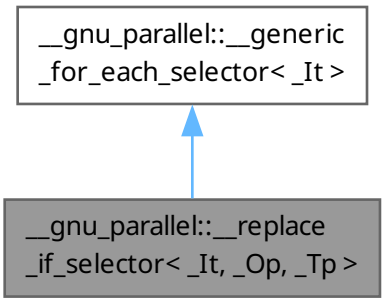
6.51.1 Detailed Description

Specialization for an empty typelist.
The documentation for this struct was generated from the following file:

- tr2/type_traits

6.52 __gnu_parallel::__replace_if_selector< _It, _Op, _Tp > Struct Template Reference

`#include <for_each_selectors.h>`
Inheritance diagram for __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >:



Public Member Functions

- __replace_if_selector (const _Tp &__new_val)
- bool operator() (_Op &__o, _It __i)

Public Attributes

- const _Tp & __new_val
- _It _M_finish_iterator

6.52.1 Detailed Description

template<typename _It, typename _Op, typename _Tp>
struct __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >

std::replace() selector.

6.52.2 Constructor & Destructor Documentation

__replace_if_selector()

```
template<typename _It, typename _Op, typename _Tp>
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__replace_if_selector (
            const _Tp & __new_val)  [inline], [explicit]
```

Constructor.

Parameters

__new_val	Value to replace with.

References __new_val.

6.52.3 Member Function Documentation

operator()()

```
template<typename _It, typename _Op, typename _Tp>
bool __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::operator() (
            _Op & __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_o	Operator.
_i	iterator referencing object.

References __new_val.

6.52.4 Member Data Documentation

__new_val

```
template<typename _It, typename _Op, typename _Tp>
const _Tp& __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >::__new_val
```

Value to replace with.
Referenced by __replace_if_selector(), and operator()().

_M_finish_iterator

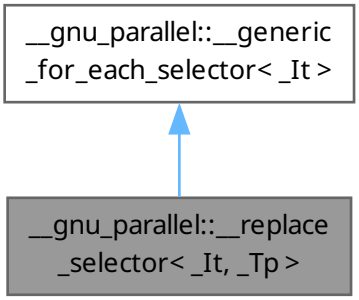
```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.53 __gnu_parallel::__replace_selector< _It, _Tp > Struct Template Reference

`#include <for_each_selectors.h>`
Inheritance diagram for __gnu_parallel::__replace_selector< _It, _Tp >:



Public Member Functions

- __replace_selector (const _Tp &__new_val)
- bool operator() (_Tp &__v, _It __i)

Public Attributes

- const _Tp & __new_val
- _It _M_finish_iterator

6.53.1 Detailed Description

template<typename _It, typename _Tp>
struct __gnu_parallel::__replace_selector< _It, _Tp >

std::replace() selector.

6.53.2 Constructor & Destructor Documentation

__replace_selector()

```
template<typename _It, typename _Tp>
__gnu_parallel::__replace_selector< _It, _Tp >::__replace_selector (
            const _Tp & __new_val)  [inline], [explicit]
```

Constructor.

Parameters

__new_val	Value to replace with.

References __new_val.

6.53.3 Member Function Documentation

operator()()

```
template<typename _It, typename _Tp>
bool __gnu_parallel::__replace_selector< _It, _Tp >::operator() (
            _Tp & __v,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _v	Current value.
_↩ _i	iterator referencing object.

References __new_val.

6.53.4 Member Data Documentation

__new_val

```
template<typename _It, typename _Tp>
const _Tp& __gnu_parallel::__replace_selector< _It, _Tp >::__new_val
```

Value to replace with.
Referenced by __replace_selector(), and operator()().

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.54 __gnu_cxx::__scoped_lock Class Reference

```
#include <concurrence.h>
```

Public Types

- typedef __mutex **__mutex_type**

Public Member Functions

- **__scoped_lock** (__mutex_type &__name)

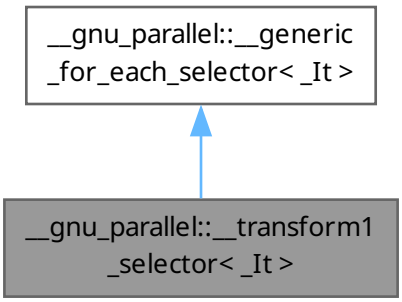
6.54.1 Detailed Description

Scoped lock idiom.
The documentation for this class was generated from the following file:

- concurrence.h

6.55 __gnu_parallel::__transform1_selector< _It > Struct Template Reference

`#include <for_each_selectors.h>`
Inheritance diagram for __gnu_parallel::__transform1_selector< _It >:



Public Member Functions

- template<typename _Op>
  bool operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

6.55.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__transform1_selector< _It >

std::transform() __selector, one input sequence variant.

6.55.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__transform1_selector< _It >::operator() (
            _Op & __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator.
_↩ _i	iterator referencing object.

6.55.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

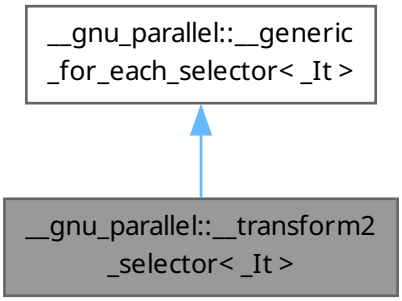
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.56 __gnu_parallel::__transform2_selector< _It > Struct Template Reference

```
#include <for_each_selectors.h>
```

Inheritance diagram for __gnu_parallel::__transform2_selector< _It >:



Public Member Functions

- template<typename _Op>
  bool operator() (_Op &__o, _It __i)

Public Attributes

- _It _M_finish_iterator

6.56.1 Detailed Description

template<typename _It>
struct __gnu_parallel::__transform2_selector< _It >

std::transform() __selector, two input sequences variant.

6.56.2 Member Function Documentation

operator()()

```
template<typename _It>
template<typename _Op>
bool __gnu_parallel::__transform2_selector< _It >::operator() (
            _Op & __o,
            _It __i)  [inline]
```

Functor execution.

Parameters

_↩ _o	Operator.
_↩ _i	iterator referencing object.

6.56.3 Member Data Documentation

_M_finish_iterator

```
template<typename _It>
_It __gnu_parallel::__generic_for_each_selector< _It >::_M_finish_iterator  [inherited]
```

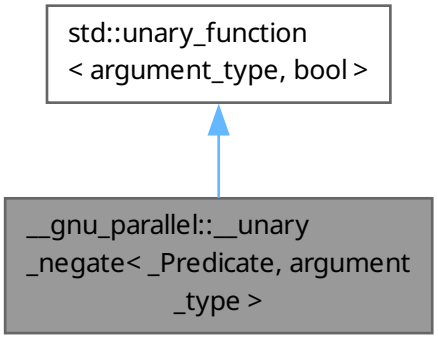
_Iterator on last element processed; needed for some algorithms (e. g. std::transform()).
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.57 __gnu_parallel::__unary_negate< _Predicate, argument_type > Class Template Reference

```
#include <base.h>
```

Inheritance diagram for __gnu_parallel::__unary_negate< _Predicate, argument_type >:



Public Types

- typedef argument_type argument_type
- typedef bool result_type

Public Member Functions

- **__unary_negate** (const _Predicate &__x)
- bool **operator()** (const argument_type &__x)

Protected Attributes

- _Predicate **_M_pred**

6.57.1 Detailed Description

template<typename _Predicate, typename argument_type>
class __gnu_parallel::__unary_negate< _Predicate, argument_type >

Similar to std::unary_negate, but giving the argument types explicitly.

6.57.2 Member Typedef Documentation

argument_type

`typedef argument_type std::unary_function< argument_type, bool >::argument_type [inherited]`
`argument_type` is the type of the argument

result_type

`typedef bool std::unary_function< argument_type, bool >::result_type [inherited]`
`result_type` is the return type
The documentation for this class was generated from the following file:

- base.h

6.58 __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > Class Template Reference

`#include <vstring.h>`

Public Types

- typedef _Alloc **allocator_type**
- typedef __gnu_cxx::__normal_iterator< const_pointer, __versa_string > **const_iterator**
- typedef _CharT_alloc_traits::const_pointer **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _CharT_alloc_type::difference_type **difference_type**
- typedef __gnu_cxx::__normal_iterator< pointer, __versa_string > **iterator**
- typedef _CharT_alloc_traits::pointer **pointer**
- typedef value_type & **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _CharT_alloc_type::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- __versa_string (__versa_string &&__str) noexcept
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  __versa_string (_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())
- __versa_string (const __versa_string &__str)
- __versa_string (const __versa_string &__str, size_type __pos, size_type __n, const _Alloc &__a)
- __versa_string (const __versa_string &__str, size_type __pos, size_type __n=npos)
- __versa_string (const _Alloc &__a=_Alloc()) noexcept
- __versa_string (const _CharT *__s, const _Alloc &__a=_Alloc())
- __versa_string (const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())
- __versa_string (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())
- __versa_string (std::initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())
- ~__versa_string () noexcept
- template<typename _InputIterator>
  __versa_string< _CharT, _Traits, _Alloc, _Base > & **_M_replace_dispatch** (const_iterator __i1, const_iterator __i2, _InputIterator __k1, _InputIterator __k2, std::__false_type)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  __versa_string & append (_InputIterator __first, _InputIterator __last)
- __versa_string & append (const __versa_string &__str)
- __versa_string & append (const __versa_string &__str, size_type __pos, size_type __n)
- __versa_string & append (const _CharT *__s)
- __versa_string & append (const _CharT *__s, size_type __n)
- __versa_string & append (size_type __n, _CharT __c)
- __versa_string & append (std::initializer_list< _CharT > __l)
- __versa_string & assign (__versa_string &&__str) noexcept
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  __versa_string & assign (_InputIterator __first, _InputIterator __last)
- __versa_string & assign (const __versa_string &__str)
- __versa_string & assign (const __versa_string &__str, size_type __pos, size_type __n)
- __versa_string & assign (const _CharT *__s)

- __versa_string & assign (const _CharT *__s, size_type __n)
- __versa_string & assign (size_type __n, _CharT __c)
- __versa_string & assign (std::initializer_list< _CharT > __l)
- reference at (size_type __n)
- const_reference at (size_type __n) const
- const_reference back () const noexcept
- reference back () noexcept
- const_iterator begin () const noexcept
- iterator begin () noexcept
- const _CharT * c_str () const noexcept
- size_type capacity () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- int compare (const __versa_string &__str) const
- int compare (const _CharT *__s) const
- int compare (size_type __pos, size_type __n, const __versa_string &__str) const
- int compare (size_type __pos, size_type __n1, const _CharT *__s) const
- int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- int compare (size_type __pos1, size_type __n1, const __versa_string &__str, size_type __pos2, size_type __n2) const
- size_type copy (_CharT *__s, size_type __n, size_type __pos=0) const
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- const _CharT * data () const noexcept
- bool empty () const noexcept
- const_iterator end () const noexcept
- iterator end () noexcept
- iterator erase (const_iterator __first, const_iterator __last)
- iterator erase (const_iterator __position)
- __versa_string & erase (size_type __pos=0, size_type __n=npos)
- size_type find (_CharT __c, size_type __pos=0) const noexcept
- size_type find (const __versa_string &__str, size_type __pos=0) const noexcept
- size_type find (const _CharT *__s, size_type __pos, size_type __n) const
- size_type find (const _CharT *__s, size_type __pos=0) const
- size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_not_of (const __versa_string &__str, size_type __pos=0) const noexcept
- size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const
- size_type find_first_not_of (const _CharT *__s, size_type __pos=0) const
- size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_of (const __versa_string &__str, size_type __pos=0) const noexcept
- size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const
- size_type find_first_of (const _CharT *__s, size_type __pos=0) const
- size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept
- size_type find_last_not_of (const __versa_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const
- size_type find_last_not_of (const _CharT *__s, size_type __pos=npos) const
- size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- size_type find_last_of (const __versa_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const
- size_type find_last_of (const _CharT *__s, size_type __pos=npos) const

- const_reference front () const noexcept
- reference front () noexcept
- allocator_type get_allocator () const noexcept
- iterator insert (const_iterator __p, _CharT __c)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator insert (const_iterator __p, _InputIterator __beg, _InputIterator __end)
- iterator insert (const_iterator __p, size_type __n, _CharT __c)
- iterator insert (const_iterator __p, std::initializer_list< _CharT > __l)
- __versa_string & insert (size_type __pos, const _CharT *__s)
- __versa_string & insert (size_type __pos, const _CharT *__s, size_type __n)
- __versa_string & insert (size_type __pos, size_type __n, _CharT __c)
- __versa_string & insert (size_type __pos1, const __versa_string &__str)
- __versa_string & insert (size_type __pos1, const __versa_string &__str, size_type __pos2, size_type __n)
- size_type length () const noexcept
- size_type max_size () const noexcept
- __versa_string & operator+= (_CharT __c)
- __versa_string & operator+= (const __versa_string &__str)
- __versa_string & operator+= (const _CharT *__s)
- __versa_string & operator+= (std::initializer_list< _CharT > __l)
- __versa_string & operator= (__versa_string &&__str) noexcept
- __versa_string & operator= (_CharT __c)
- __versa_string & operator= (const __versa_string &__str)
- __versa_string & operator= (const _CharT *__s)
- __versa_string & operator= (std::initializer_list< _CharT > __l)
- const_reference operator[ ] (size_type __pos) const noexcept
- reference operator[ ] (size_type __pos) noexcept
- void pop_back ()
- void push_back (_CharT __c)
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rend () const noexcept
- reverse_iterator rend () noexcept
- __versa_string & **replace** (const_iterator __i1, const_iterator __i2, _CharT *__k1, _CharT *__k2)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  __versa_string & replace (const_iterator __i1, const_iterator __i2, _InputIterator __k1, _InputIterator __k2)
- __versa_string & replace (const_iterator __i1, const_iterator __i2, const __versa_string &__str)
- __versa_string & **replace** (const_iterator __i1, const_iterator __i2, const _CharT *__k1, const _CharT *__k2)
- __versa_string & replace (const_iterator __i1, const_iterator __i2, const _CharT *__s)
- __versa_string & replace (const_iterator __i1, const_iterator __i2, const _CharT *__s, size_type __n)
- __versa_string & **replace** (const_iterator __i1, const_iterator __i2, const_iterator __k1, const_iterator __k2)
- __versa_string & **replace** (const_iterator __i1, const_iterator __i2, iterator __k1, iterator __k2)
- __versa_string & replace (const_iterator __i1, const_iterator __i2, size_type __n, _CharT __c)
- __versa_string & replace (const_iterator __i1, const_iterator __i2, std::initializer_list< _CharT > __l)
- __versa_string & replace (size_type __pos, size_type __n, const __versa_string &__str)
- __versa_string & replace (size_type __pos, size_type __n1, const _CharT *__s)
- __versa_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)
- __versa_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- __versa_string & replace (size_type __pos1, size_type __n1, const __versa_string &__str, size_type __pos2, size_type __n2)
- void reserve (size_type __res_arg=0)
- void resize (size_type __n)

- void resize (size_type __n, _CharT __c)
- size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- size_type rfind (const __versa_string &__str, size_type __pos=npos) const noexcept
- size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const
- size_type rfind (const _CharT *__s, size_type __pos=npos) const
- void shrink_to_fit () noexcept
- size_type size () const noexcept
- __versa_string substr (size_type __pos=0, size_type __n=npos) const
- void swap (__versa_string &__s) noexcept

Static Public Attributes

- static const size_type npos

6.58.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >

Template class __versa_string.
Data structure managing sequences of characters and character-like objects.

6.58.2 Constructor & Destructor Documentation

__versa_string() [1/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            const _Alloc & __a = _Alloc())  [inline], [explicit], [noexcept]
```

Construct an empty string using allocator *a*.
Referenced by compare(), compare(), and __gnu_cxx::__versa_string< char >::substr().

__versa_string() [2/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Construct string with copy of value of *__str*.

Parameters

__str	Source string.

__versa_string() [3/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            __versa_string< _CharT, _Traits, _Alloc, _Base > && __str)  [inline], [noexcept]
```

String move constructor.

Parameters

__str	Source string.

The newly-constructed string contains the exact contents of *__str*. The contents of *__str* are a valid, but unspecified string.

__versa_string() [4/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            std::initializer_list< _CharT > __l,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string from an initializer list.

Parameters

_↩ _l	std::initializer_list of characters.
_↩ _a	Allocator to use (default is default allocator).

__versa_string() [5/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos,
            size_type __n = npos)  [inline]
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy (default remainder).

__versa_string() [6/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos,
```

```
            size_type __n,
            const _Alloc & __a)  [inline]
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy.
__a	Allocator to use.

__versa_string() [7/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            const _CharT * __s,
            size_type __n,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string initialized by a character array.

Parameters

_↩ _s	Source character array.
_↩ _n	Number of characters to copy.
_↩ _a	Allocator to use (default is default allocator).

NB: *__s* must have at least *__n* characters, '\0' has no special meaning.

__versa_string() [8/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            const _CharT * __s,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string as copy of a C string.

Parameters

_↩ _s	Source C string.
_↩ _a	Allocator to use (default is default allocator).

__versa_string() [9/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            size_type __n,
            _CharT __c,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string as multiple characters.

Parameters

_↩ _n	Number of characters.
_↩ _c	Character to use.
_↩ _a	Allocator to use (default is default allocator).

__versa_string() [10/10]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::__versa_string (
            _InputIterator __beg,
            _InputIterator __end,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string as copy of a range.

Parameters

__beg	Start of range.
__end	End of range.
__a	Allocator to use (default is default allocator).

~__versa_string()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::~__versa_string ()  [inline], [noexcept]
```

Destroy the string instance.

6.58.3 Member Function Documentation

append() [1/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
```

```
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Append a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [first,last) to this string.

append() [2/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

Referenced by __gnu_cxx::__versa_string< char >::append(), std::getline(), __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::__versa_string< char >::operator+=(), __gnu_cxx::__versa_string< char >::operator+=(), __gnu_cxx::__versa_string< char >::operator+=(), and resize().

append() [3/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos,
            size_type __n)  [inline]
```

Append a substring.

Parameters

__str	The string to append.
__pos	Index of the first character of str to append.
__n	The number of characters to append.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *pos* is not a valid index.

This function appends *__n* characters from *__str* starting at *__pos* to this string. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is appended.

append() [4/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            const _CharT * __s)  [inline]
```

Append a C string.

Parameters

_↩ _s	The C string to append.

Returns

Reference to this string.

append() [5/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            const _CharT * __s,
            size_type __n)  [inline]
```

Append a C substring.

Parameters

_↩ _s	The C string to append.
_↩ _n	The number of characters to append.

Returns

Reference to this string.

append() [6/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            size_type __n,
            _CharT __c)  [inline]
```

Append multiple characters.

Parameters

_↩ _*n*	The number of characters to append.
_↩ _*c*	The character to use.

Returns

Reference to this string.

Appends n copies of c to this string.

append() [7/7]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::append (
            std::initializer_list< _CharT > __l)  [inline]
```

Append an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to append.

Returns

Reference to this string.

assign() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            __versa_string< _CharT, _Traits, _Alloc, _Base > && __str)  [inline], [noexcept]
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of __*str*. __*str* is a valid, but unspecified string.

assign() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Set value to a range of characters.

Parameters

__*first*	Iterator referencing the first character to append.
__*last*	Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range [first,last).

assign() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Set value to contents of another string.

Parameters

__*str*	Source string to use.

Returns

Reference to this string.

Referenced by __gnu_cxx::__versa_string< char >::assign(), __gnu_cxx::__versa_string< char >::operator=(), __gnu_cxx::__versa_string< char >::operator=(), __gnu_cxx::__versa_string< char >::operator=(), and __gnu_cxx::__versa_string< ch

assign() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos,
            size_type __n)  [inline]
```

Set value to a substring of a string.

Parameters

__str	The string to use.
__pos	Index of the first character of str.
__n	Number of characters to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *__pos* is not a valid index.

This function sets this string to the substring of *__str* consisting of *__n* characters at *__pos*. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is used.

assign() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            const _CharT * __s)  [inline]
```

Set value to contents of a C string.

Parameters

_↩ _s	The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of *__s*. The data is copied, so there is no dependence on *__s* once the function returns.

assign() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            const _CharT * __s,
            size_type __n)  [inline]
```

Set value to a C substring.

Parameters

_↩ _s	The C string to use.
_↩ _n	Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first __n characters of __s. If __n is is larger than the number of available characters in __s, the remainder of __s is used.

assign() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            size_type __n,
            _CharT __c)  [inline]
```

Set value to multiple characters.

Parameters

_↩ _n	Length of the resulting string.
_↩ _c	The character to use.

Returns

Reference to this string.

This function sets the value of this string to __n copies of character __c.

assign() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::assign (
            std::initializer_list< _CharT > __l)  [inline]
```

Set value to an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ l	The initializer_list of characters to assign.

Returns

Reference to this string.

at() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at (
            size_type __n)  [inline]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If __n is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails. Success results in unsharing the string.

at() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::at (
            size_type __n) const  [inline]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If __n is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

back() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back () const  [inline],
[noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

back() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::back ()  [inline], [noexcept]
```

Returns a read/write reference to the data at the last element of the string.

begin() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

begin() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::begin ()  [inline], [noexcept]
```

Returns a read/write iterator that points to the first character in the string. Unshares the string.

Referenced by __gnu_cxx::__versa_string< char >::crend(), __gnu_cxx::__versa_string< char >::rend(), and __gnu_cxx::__versa_string< char >::rend().

c_str()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const _CharT * __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::c_str () const  [inline],
[noexcept]
```

Return const pointer to null-terminated contents.

This is a handle to internal data. Do not modify or dire things may happen.

capacity()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::capacity () const  [inline],
[noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

Referenced by __gnu_cxx::__versa_string< char >::push_back(), and __gnu_cxx::__versa_string< char >::shrink_to_fit().

cbegin()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

cend()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::cend () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

clear()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::clear ()  [inline], [noexcept]
```

Erases the string, making it empty.

compare() [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) const  [inline]
```

Compare to a string.

Parameters

__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *__str*, 0 if their values are equivalent, or > 0 if this string is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of size() and str.size(). The function then compares the two strings by calling traits::compare(data(), str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

Referenced by __gnu_cxx::operator<(), __gnu_cxx::operator<(), __gnu_cxx::operator<(), __gnu_cxx::operator<=(), __gnu_cxx::operator<=(), __gnu_cxx::operator<=(), __gnu_cxx::operator>(), __gnu_cxx::operator>(), __gnu_cxx::operator>(), __gnu_cxx::operator>=(), __gnu_cxx::operator>=(), and __gnu_cxx::operator>=().

compare() [2/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
            const _CharT * __s) const
```

Compare to a C string.

Parameters

__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before __*s*, 0 if their values are equivalent, or > 0 if this string is ordered after __*s*. Determines the effective length rlen of the strings to compare as the smallest of size() and the length of a string constructed from __*s*. The function then compares the two strings by calling traits::compare(data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
References std::min(), and size().

compare() [3/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
            size_type __pos,
            size_type __n,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str) const
```

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the __*n* characters starting at __*pos*. Returns an integer < 0 if the substring is ordered before __*str*, 0 if their values are equivalent, or > 0 if the substring is ordered after __*str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and __*str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
References __versa_string(), data(), std::min(), and size().

compare() [4/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s) const
```

Compare substring to a C string.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.

__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *__s*. The function then compares the two string by calling traits::compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
References std::min().

compare() [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2) const
```

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Form a string from the first *__n2* characters of *__s*. Returns an integer < 0 if this substring is ordered before the string from *__s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__n2*. The function then compares the two strings by calling traits::↩ compare(substring.data(),__s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
NB: __s must have at least n2 characters, *\0* has no special meaning.
References std::min().

compare() [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
int __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::compare (
            size_type __pos1,
```

```
            size_type __n1,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos2,
            size_type __n2) const
```

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer < 0 if this substring is ordered before the substring of *__str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *__str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
References __versa_string(), data(), and std::min().

copy()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::copy (
            _CharT * __s,
            size_type __n,
            size_type __pos = 0) const
```

Copy substring into C string.

Parameters

__s	C string to copy value into.
__n	Number of characters to copy.
__pos	Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

std::out_of_range	If pos > size().

Copies up to *__n* characters starting at *__pos* into the C string *s*. If *__pos* is greater than size(), out_of_range is thrown.

crbegin()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crbegin ()
const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

crend()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::crend ()
const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

data()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const _CharT * __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::data () const  [inline],
[noexcept]
```

Return const pointer to contents.

This is a handle to internal data. Do not modify or dire things may happen.

Referenced by compare(), compare(), __gnu_cxx::__versa_string< char >::compare(), __gnu_cxx::__versa_string< char >::find(), __gnu_cxx::__versa_string< char >::find_first_not_of(), __gnu_cxx::__versa_string< char >::find_first_of(), __gnu_cxx::__versa_string< __gnu_cxx::operator==(), and __gnu_cxx::operator==().

empty()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
bool __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::empty () const  [inline], [nodiscard],
[noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

end() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

end() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::end () [inline], [noexcept]
```

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

Referenced by __gnu_cxx::__versa_string< char >::crbegin(), __gnu_cxx::__versa_string< char >::rbegin(), and __gnu_cxx::__versa_string< char >::rbegin().

erase() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
            const_iterator __first,
            const_iterator __last) [inline]
```

Remove a range of characters.

Parameters

__first	Iterator referencing the first character to remove.
__last	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

erase() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
            const_iterator __position) [inline]
```

Remove one character.

Parameters

__position	Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *__position* from this string. The value of the string doesn't change if an error is thrown.

erase() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::erase (
```

```
            size_type __pos = 0,
            size_type __n = npos)  [inline]
```

Remove characters.

Parameters

__pos	Index of first character to remove (default 0).
__n	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* is beyond the end of this string.

Removes *__n* characters from this string starting at *__pos*. The length of the string is reduced by *__n*. If there are < *__n* characters to remove, the remainder of the string is truncated. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::getline().

find() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find (
            _CharT __c,
            size_type __pos = 0) const  [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

find() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (
```

```
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const
```

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *__s* to search for.

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.
References npos, and size().
Referenced by __gnu_cxx::__versa_string< char >::find(), __gnu_cxx::__versa_string< char >::find(), and __gnu_cxx::__versa_string<

find() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find (
            const _CharT * __s,
            size_type __pos = 0) const  [inline]
```

Find position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

find_first_not_of() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find_first_not_of (
            _CharT __c,
            size_type __pos = 0) const  [noexcept]
```

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

find_first_not_of() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by __gnu_cxx::__versa_string< char >::find_first_not_of(), and __gnu_cxx::__versa_string< char >::find_first_not_of().

find_first_not_of() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find_first_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const
```

Find position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from s to consider.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

find_first_not_of() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_not_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline]
```

Find position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
            _CharT __c,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to find(c, pos).

find_first_of() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by __gnu_cxx::__versa_string< char >::find_first_of(), and __gnu_cxx::__versa_string< char >::find_first_of().

find_first_of() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find_first_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const
```

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

find_first_of() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_first_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline]
```

Find position of a character of C string.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find_last_not_of (
            _CharT __c,
            size_type __pos = npos) const  [noexcept]
```

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

find_last_not_of() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by __gnu_cxx::__versa_string< char >::find_last_not_of(), and __gnu_cxx::__versa_string< char >::find_last_not_of().

find_last_not_of() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find_last_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const
```

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

find_last_not_of() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_not_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline]
```

Find last position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
            _CharT __c,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from __*pos*, searches backward for __*c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to rfind(c, pos).

find_last_of() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character of string.

Parameters

__*str*	String containing characters to locate.
__*pos*	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from __*pos*, searches backward for one of the characters of __*str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by __gnu_cxx::__versa_string< char >::find_last_of(), and __gnu_cxx::__versa_string< char >::find_last_of().

find_last_of() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::find_last_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const
```

Find last position of a character of C substring.

Parameters

__*s*	C string containing characters to locate.
__*pos*	Index of character to search back from.
__*n*	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

find_last_of() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::find_last_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline]
```

Find last position of a character of C string.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

front() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front () const  [inline],
[noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

front() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::front ()  [inline], [noexcept]
```

Returns a read/write reference to the data at the first element of the string.

get_allocator()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
allocator_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::get_allocator ()
const  [inline], [noexcept]
```

Return copy of allocator used to construct this string.

insert() [1/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            const_iterator __p,
            _CharT __c)  [inline]
```

Insert one character.

Parameters

_↩ _p	Iterator referencing position in string to insert at.
_↩ _c	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts character __c at position referenced by __p. If adding character causes the length to exceed max_size(), length_error is thrown. If __p is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [2/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            const_iterator __p,
            _InputIterator __beg,
            _InputIterator __end)  [inline]
```

Insert a range of characters.

Parameters

__p	Const_iterator referencing location in string to insert at.
__beg	Start of range.
__end	End of range.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts characters in range [beg,end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [3/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            const_iterator __p,
            size_type __n,
            _CharT __c)  [inline]
```

Insert multiple characters.

Parameters

__p	Const_iterator referencing location in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts *__n* copies of character *__c* starting at the position referenced by iterator *__p*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.
Referenced by __gnu_cxx::__versa_string< char >::insert().

insert() [4/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            const_iterator __p,
            std::initializer_list< _CharT > __l)  [inline]
```

Insert an initializer_list of characters.

Parameters

_↩ _p	Const_iterator referencing location in string to insert at.
_↩ _l	The initializer_list of characters to insert.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

insert() [5/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            size_type __pos,
            const _CharT * __s)  [inline]
```

Insert a C string.

Parameters

__pos	Iterator referencing location in string to insert at.
__s	The C string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts the first *__n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [6/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            size_type __pos,
            const _CharT * __s,
            size_type __n)  [inline]
```

Insert a C substring.

Parameters

__pos	Iterator referencing location in string to insert at.
__s	The C string to insert.
__n	The number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts the first *__n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [7/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            size_type __pos,
            size_type __n,
            _CharT __c)  [inline]
```

Insert multiple characters.

Parameters

__pos	Index in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts *__n* copies of character *__c* starting at index *__pos*. If adding characters causes the length to exceed max_↩ size(), length_error is thrown. If *__pos* > length(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [8/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            size_type __pos1,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Insert value of a string.

Parameters

__pos1	Iterator referencing location in string to insert at.
__str	The string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts value of *__str* starting at *__pos1*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [9/9]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::insert (
            size_type __pos1,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos2,
            size_type __n)  [inline]
```

Insert a substring.

Parameters

__pos1	Iterator referencing location in string to insert at.
__str	The string to insert.
__pos2	Start of characters in str to insert.
__n	Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos1* > size() or *__pos2* > *__str.size()*.

Starting at *__pos1*, insert *__n* character of *__str* beginning with *__pos2*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos1* is beyond the end of this string or *__pos2* is beyond the end of *__str*, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

length()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::length () const  [inline],
[noexcept]
```

Returns the number of characters in the string, not including any null-termination.

max_size()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::max_size () const  [inline],
[noexcept]
```

Returns the size() of the largest possible string.
Referenced by std::getline().

operator+=() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
            _CharT __c)  [inline]
```

Append a character.

Parameters

__c	The character to append.

Returns

Reference to this string.

operator+=() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

operator+=() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
            const _CharT * __s)  [inline]
```

Append a C string.

Parameters

_↩ _s	The C string to append.

Returns

Reference to this string.

operator+=() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator+= (
            std::initializer_list< _CharT > __l)  [inline]
```

Append an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to be appended.

Returns

Reference to this string.

operator=() [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
            __versa_string< _CharT, _Traits, _Alloc, _Base > && __str)  [inline], [noexcept]
```

String move assignment operator.

Parameters

__str	Source string.

The contents of *__str* are moved into this string (without copying). *__str* is a valid, but unspecified string.

operator=() [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
            _CharT __c)  [inline]
```

Set value to string of length 1.

Parameters

_↩ *_c*	Source character.

Assigning to a character makes this string length 1 and (*this)[0] == *__c*.

operator=() [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Assign the value of *str* to this string.

Parameters

__str	Source string.

operator=() [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
            const _CharT * __s)  [inline]
```

Copy contents of *__s* into this string.

Parameters

_↩ _*s*	Source null-terminated string.

operator=() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator= (
            std::initializer_list< _CharT > __l)  [inline]
```

Set value to string constructed from initializer list.

Parameters

↩ _↩ ↩ _↩ *l*	std::initializer_list.

operator[]() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] (
            size_type __pos) const  [inline], [noexcept]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)
Referenced by __gnu_cxx::__versa_string< char >::back(), __gnu_cxx::__versa_string< char >::back(), __gnu_cxx::__versa_string< ch and __gnu_cxx::__versa_string< char >::front().

operator[]() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reference __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::operator[] (
            size_type __pos) [inline], [noexcept]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().) Unshares the string.

pop_back()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::pop_back () [inline]
```

Remove the last character.
The string must be non-empty.

push_back()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::push_back (
            _CharT __c) [inline]
```

Append a single character.

Parameters

__c	Character to append.

Referenced by __gnu_cxx::operator+(), __gnu_cxx::operator+(), and __gnu_cxx::__versa_string< char >::operator+=().

rbegin() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin ()
const [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

rbegin() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rbegin ()  [inline],
[noexcept]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

rend() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const_reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend () const
[inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

rend() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
reverse_iterator __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rend ()  [inline],
[noexcept]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

replace() [1/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            const_iterator __i1,
            const_iterator __i2,
            _InputIterator __k1,
            _InputIterator __k2)  [inline]
```

Replace range of characters with range.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__k1	Iterator referencing start of range to insert.
__k2	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, characters in the range [k1,k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [2/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            const_iterator __i1,
            const_iterator __i2,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Replace range of characters with string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__str	String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, the value of *__str* is inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [3/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            const_iterator __i1,
            const_iterator __i2,
            const _CharT * __s)  [inline]
```

Replace range of characters with C string.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _s	C string value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, the characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [4/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            const_iterator __i1,
            const_iterator __i2,
            const _CharT * __s,
            size_type __n)  [inline]
```

Replace range of characters with C substring.

Parameters

_↩ _*i1*	Iterator referencing start of range to replace.
_↩ _*i2*	Iterator referencing end of range to replace.
_↩ _*s*	C string value to insert.
_↩ _*n*	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, the first *n* characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [5/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            const_iterator __i1,
            const_iterator __i2,
```

```
            size_type __n,
            _CharT __c)  [inline]
```

Replace range of characters with multiple characters.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _n	Number of characters to insert.
_↩ _c	Character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, __n copies of __c are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [6/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            const_iterator __i1,
            const_iterator __i2,
            std::initializer_list< _CharT > __l)  [inline]
```

Replace range of characters with initializer_list.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _l	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [i1,i2). In place, characters in the range [k1,k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [7/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            size_type __pos,
            size_type __n,
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str)  [inline]
```

Replace characters with value from another string.

Parameters

__pos	Index of first character to replace.
__n	Number of characters to be replaced.
__str	String to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* is beyond the end of this string.
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos+n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.
Referenced by __gnu_cxx::__versa_string< char >::append(), __gnu_cxx::__versa_string< char >::assign(), __gnu_cxx::__versa_string< char >::insert(), __gnu_cxx::__versa_string< char >::insert(), __gnu_cxx::__versa_string< char >::insert(), __gnu_cxx::__versa_string< char >::insert(), __gnu_cxx::__versa_string< char >::insert(), __gnu_cxx::__versa_string< char >::replace(), __gnu_cxx::__versa_string< char >::replace(), __gnu_cxx::__versa_string< char >::rep, __gnu_cxx::__versa_string< char >::replace(), __gnu_cxx::__versa_string< char >::replace(), __gnu_cxx::__versa_string< char >::rep, and __gnu_cxx::__versa_string< char >::replace().

replace() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
```

```
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            size_type __pos,
            size_type __n1,
            const _CharT * __s)  [inline]
```

Replace characters with value of a C string.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, the characters of *__s* are inserted. If *pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2)  [inline]
```

Replace characters with value of a C substring.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.
__n2	Number of characters from *__s* to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, the first *__n2* characters of *__s* are inserted, or all of *__s* if *__n2* is too large. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            size_type __pos,
            size_type __n1,
            size_type __n2,
            _CharT __c)  [inline]
```

Replace characters with multiple characters.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__n2	Number of characters to insert.
__c	Character to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, *__n2* copies of *__c* are inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string & __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::replace (
            size_type __pos1,
```

```
        size_type __n1,
        const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
        size_type __pos2,
        size_type __n2)  [inline]
```

Replace characters with value from another string.

Parameters

__pos1	Index of first character to replace.
__n1	Number of characters to be replaced.
__str	String to insert.
__pos2	Index of first character of str to use.
__n2	Number of characters from str to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size() or *__pos2* > str.size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos1,pos1 + n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

reserve()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::reserve (
        size_type __res_arg = 0)  [inline]
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.
Referenced by __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::operator+(), and __gnu_cxx::__versa_string< char >::shrink

resize() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize (
            size_type __n)  [inline]
```

Resizes the string to the specified number of characters.

Parameters

_↩ _*n*	Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

resize() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::resize (
            size_type __n,
            _CharT __c)
```

Resizes the string to the specified number of characters.

Parameters

_↩ _*n*	Number of characters the string should contain.
_↩ _*c*	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to __*c*.
References append(), and size().
Referenced by __gnu_cxx::__versa_string< char >::resize().

rfind() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::rfind (
            _CharT __c,
            size_type __pos = npos) const  [noexcept]
```

Find last position of a character.

Parameters

__*c*	Character to locate.
__*pos*	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and size().

rfind() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (
            const __versa_string< _CharT, _Traits, _Alloc, _Base > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.
Referenced by __gnu_cxx::__versa_string< char >::find_last_of(), __gnu_cxx::__versa_string< char >::rfind(), and __gnu_cxx::__versa_string< char >::rfind().

rfind() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base >::size_type __gnu_cxx::__versa_string< _CharT, ↩
_Traits, _Alloc, _Base >::rfind (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const
```

Find last position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from __pos, searches backward for the first __n characters in __s within this string. If found, returns the index where it begins. If not found, returns npos.
References std::min(), npos, and size().

rfind() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::rfind (
            const _CharT * __s,
            size_type __pos = npos) const  [inline]
```

Find last position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from __pos, searches backward for the value of __s within this string. If found, returns the index where it begins. If not found, returns npos.

shrink_to_fit()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::shrink_to_fit ()  [inline],
[noexcept]
```

A non-binding request to reduce capacity() to size().

size()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::size () const  [inline],
[noexcept]
```

Returns the number of characters in the string, not including any null-termination.
Referenced by __gnu_cxx::__versa_string< char >::append(), __gnu_cxx::__versa_string< char >::append(), __gnu_cxx::__versa_string< char >::assign(), __gnu_cxx::__versa_string< char >::assign(), __gnu_cxx::__versa_string< char >::assig __gnu_cxx::__versa_string< char >::assign(), __gnu_cxx::__versa_string< char >::at(), __gnu_cxx::__versa_string< char >::at(), __gnu_cxx::__versa_string< char >::back(), __gnu_cxx::__versa_string< char >::back(), __gnu_cxx::__versa_string< char >::cend(), compare(), compare(), __gnu_cxx::__versa_string< char >::compare(), __gnu_cxx::__versa_string< char >::empty(), __gnu_cxx::__versa_string< char >::end(), __gnu_cxx::__versa_string< char >::end(), find(), find(), __gnu_cxx::__versa_string< char > find_first_not_of(), find_first_not_of(), __gnu_cxx::__versa_string< char >::find_first_not_of(), find_first_of(), __gnu_cxx::__versa_string< find_last_not_of(), find_last_not_of(), find_last_of(), __gnu_cxx::__versa_string< char >::find_last_of(), __gnu_cxx::__versa_string< cha __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::operator+(), __gnu_cxx::operator==(), __gnu_cxx::operator==(), __gnu_cxx::__versa_string< char >::operator[](), __gnu_cxx::__versa_string< char

__gnu_cxx::__versa_string< char >::pop_back(), __gnu_cxx::__versa_string< char >::push_back(), __gnu_cxx::__versa_string< char >
__gnu_cxx::__versa_string< char >::replace(), resize(), rfind(), rfind(), and __gnu_cxx::__versa_string< char >::shrink_to_fit().

substr()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
__versa_string __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::substr (
            size_type __pos = 0,
            size_type __n = npos) const  [inline]
```

Get a substring.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

std::out_of_range	If pos > size().

Construct and return a new string using the *__n* characters starting at *__pos*. If the string is too short, use the remainder of the characters. If *__pos* is beyond the end of the string, out_of_range is thrown.

swap()

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
void __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::swap (
            __versa_string< _CharT, _Traits, _Alloc, _Base > & __s)  [inline], [noexcept]
```

Swap contents with another string.

Parameters

__s	String to swap with.

Exchanges the contents of this string with that of *__s* in constant time.
Referenced by __gnu_cxx::__versa_string< char >::assign(), __gnu_cxx::__versa_string< char >::operator=(), and __gnu_cxx::swap().

6.58.4 Member Data Documentation

npos

```
template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename
> class _Base>
const size_type __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos  [static]
```

Value returned by various member functions when they fail.
Referenced by find(), find(), find_first_not_of(), find_first_not_of(), find_first_of(), find_last_not_of(), find_last_not_of(), find_last_of(), rfind(), and rfind().
The documentation for this class was generated from the following files:

- vstring.h
- vstring.tcc

6.59 __gnu_debug::_After_nth_from< _Iterator > Class Template Reference

`#include <safe_sequence.h>`

Public Member Functions

- constexpr **_After_nth_from** (const difference_type &__n, const _Iterator &__base)
- bool **operator()** (const _Iterator &__x) const

6.59.1 Detailed Description

template<typename _Iterator>
class __gnu_debug::_After_nth_from< _Iterator >

A function object that returns true when the given random access iterator is at least `n` steps away from the given iterator.
The documentation for this class was generated from the following file:

- safe_sequence.h

6.60 std::_Base_bitset< _Nw > Struct Template Reference

`#include <bitset>`

Public Types

- typedef unsigned long **_WordT**

Public Member Functions

- constexpr **_Base_bitset** (unsigned long long __val) noexcept
- template<size_t _Nb>
  constexpr bool **_M_are_all** () const noexcept
- constexpr void **_M_do_and** (const _Base_bitset< _Nw > &__x) noexcept
- constexpr size_t **_M_do_count** () const noexcept
- constexpr size_t **_M_do_find_first** (size_t) const noexcept
- constexpr size_t **_M_do_find_next** (size_t, size_t) const noexcept
- constexpr void **_M_do_flip** () noexcept
- constexpr void **_M_do_left_shift** (size_t __shift) noexcept
- constexpr void **_M_do_or** (const _Base_bitset< _Nw > &__x) noexcept
- constexpr void **_M_do_reset** () noexcept
- constexpr void **_M_do_right_shift** (size_t __shift) noexcept
- constexpr void **_M_do_set** () noexcept
- constexpr unsigned long long **_M_do_to_ullong** () const
- constexpr unsigned long **_M_do_to_ulong** () const
- constexpr void **_M_do_xor** (const _Base_bitset< _Nw > &__x) noexcept
- constexpr const _WordT * **_M_getdata** () const noexcept
- constexpr _WordT **_M_getword** (size_t __pos) const noexcept

- constexpr _WordT & **_M_getword** (size_t __pos) noexcept
- constexpr _WordT **_M_hiword** () const noexcept
- constexpr _WordT & **_M_hiword** () noexcept
- constexpr bool **_M_is_any** () const noexcept
- constexpr bool **_M_is_equal** (const _Base_bitset< _Nw > &__x) const noexcept

Static Public Member Functions

- static constexpr _WordT **_S_maskbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbyte** (size_t __pos) noexcept
- static constexpr size_t **_S_whichword** (size_t __pos) noexcept

Public Attributes

- _WordT _M_w [_Nw]

6.60.1 Detailed Description

template<size_t _Nw>
struct std::_Base_bitset< _Nw >

Base class, general case. It is a class invariant that _Nw will be nonnegative.
See documentation for bitset.

6.60.2 Member Data Documentation

_M_w

```
template<size_t _Nw>
_WordT std::_Base_bitset< _Nw >::_M_w[_Nw]
```

0 is the least significant word.
The documentation for this struct was generated from the following file:

- bitset

6.61 std::_Base_bitset< 0 > Struct Reference

```
#include <bitset>
```

Public Types

- typedef unsigned long **_WordT**

Public Member Functions

- constexpr **_Base_bitset** (unsigned long long) noexcept
- template<size_t _Nb>
  constexpr bool **_M_are_all** () const noexcept
- constexpr void **_M_do_and** (const _Base_bitset< 0 > &) noexcept
- constexpr size_t **_M_do_count** () const noexcept
- constexpr size_t **_M_do_find_first** (size_t) const noexcept
- constexpr size_t **_M_do_find_next** (size_t, size_t) const noexcept
- constexpr void **_M_do_flip** () noexcept
- constexpr void **_M_do_left_shift** (size_t) noexcept

- constexpr void **_M_do_or** (const _Base_bitset< 0 > &) noexcept
- constexpr void **_M_do_reset** () noexcept
- constexpr void **_M_do_right_shift** (size_t) noexcept
- constexpr void **_M_do_set** () noexcept
- constexpr unsigned long long **_M_do_to_ullong** () const noexcept
- constexpr unsigned long **_M_do_to_ulong** () const noexcept
- constexpr void **_M_do_xor** (const _Base_bitset< 0 > &) noexcept
- constexpr const _WordT * **_M_getdata** () const noexcept
- constexpr _WordT **_M_getword** (size_t) const noexcept
- _WordT & **_M_getword** (size_t) noexcept
- constexpr _WordT **_M_hiword** () const noexcept
- constexpr bool **_M_is_any** () const noexcept
- constexpr bool **_M_is_equal** (const _Base_bitset< 0 > &) const noexcept

Static Public Member Functions

- static constexpr _WordT **_S_maskbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbyte** (size_t __pos) noexcept
- static constexpr size_t **_S_whichword** (size_t __pos) noexcept

Public Attributes

- _WordT _M_w [_Nw]

6.61.1 Detailed Description

Base class, specialization for no storage (zero-length bitset).
See documentation for bitset.

6.61.2 Member Data Documentation

_M_w

```
_WordT std::_Base_bitset< _Nw >::_M_w[_Nw]
```

0 is the least significant word.
The documentation for this struct was generated from the following file:

- bitset

6.62 std::_Base_bitset< 1 > Struct Reference

```
#include <bitset>
```

Public Types

- typedef unsigned long **_WordT**

Public Member Functions

- constexpr **_Base_bitset** (unsigned long long __val) noexcept
- template<size_t _Nb>
  constexpr bool **_M_are_all** () const noexcept
- constexpr void **_M_do_and** (const _Base_bitset< 1 > &__x) noexcept
- constexpr size_t **_M_do_count** () const noexcept

- constexpr size_t **_M_do_find_first** (size_t __not_found) const noexcept
- constexpr size_t **_M_do_find_next** (size_t __prev, size_t __not_found) const noexcept
- constexpr void **_M_do_flip** () noexcept
- constexpr void **_M_do_left_shift** (size_t __shift) noexcept
- constexpr void **_M_do_or** (const _Base_bitset< 1 > &__x) noexcept
- constexpr void **_M_do_reset** () noexcept
- constexpr void **_M_do_right_shift** (size_t __shift) noexcept
- constexpr void **_M_do_set** () noexcept
- constexpr unsigned long long **_M_do_to_ullong** () const noexcept
- constexpr unsigned long **_M_do_to_ulong** () const noexcept
- constexpr void **_M_do_xor** (const _Base_bitset< 1 > &__x) noexcept
- constexpr const _WordT * **_M_getdata** () const noexcept
- constexpr _WordT **_M_getword** (size_t) const noexcept
- constexpr _WordT & **_M_getword** (size_t) noexcept
- constexpr _WordT **_M_hiword** () const noexcept
- constexpr _WordT & **_M_hiword** () noexcept
- constexpr bool **_M_is_any** () const noexcept
- constexpr bool **_M_is_equal** (const _Base_bitset< 1 > &__x) const noexcept

Static Public Member Functions

- static constexpr _WordT **_S_maskbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbit** (size_t __pos) noexcept
- static constexpr size_t **_S_whichbyte** (size_t __pos) noexcept
- static constexpr size_t **_S_whichword** (size_t __pos) noexcept

Public Attributes

- _WordT **_M_w**

6.62.1 Detailed Description

Base class, specialization for a single word.
See documentation for bitset.
The documentation for this struct was generated from the following file:

- bitset

6.63 __gnu_debug::_BeforeBeginHelper< _Sequence > Struct Template Reference

`#include <safe_iterator.h>`

Static Public Member Functions

- template<typename _Iterator, typename _Category>
  static bool **_S_Is** (const _Safe_iterator< _Iterator, _Sequence, _Category > &)
- template<typename _Iterator, typename _Category>
  static bool **_S_Is_Beginnest** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it)

6.63.1 Detailed Description

template<typename _Sequence>
struct __gnu_debug::_BeforeBeginHelper< _Sequence >

Helper struct to deal with sequence offering a before_begin iterator.
The documentation for this struct was generated from the following file:

- safe_iterator.h

6.64 std::_Bind< _Signature > Class Template Reference

6.64.1 Detailed Description

template<typename _Signature>
class std::_Bind< _Signature >

Type of the function object returned from bind().
The documentation for this class was generated from the following file:

- functional

6.65 std::_Bind_result< _Result, _Signature > Class Template Reference

6.65.1 Detailed Description

template<typename _Result, typename _Signature>
class std::_Bind_result< _Result, _Signature >

Type of the function object returned from bind<R>().
The documentation for this class was generated from the following file:

- functional

6.66 __gnu_cxx::__detail::_Bitmap_counter< _Tp > Class Template Reference

`#include <bitmap_allocator.h>`

Public Member Functions

- **_Bitmap_counter** (_BPVector &Rvbp, long __index=-1)
- pointer **_M_base** () const throw ()
- bool **_M_finished** () const throw ()
- std::size_t * **_M_get** () const throw ()
- _Index_type **_M_offset** () const throw ()
- void **_M_reset** (long __index=-1) throw ()
- void **_M_set_internal_bitmap** (std::size_t *__new_internal_marker) throw ()
- _Index_type **_M_where** () const throw ()
- _Bitmap_counter & **operator++** () throw ()

6.66.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::__detail::_Bitmap_counter< _Tp >

The bitmap counter which acts as the bitmap manipulator, and manages the bit-manipulation functions and the searching and identification functions on the bit-map.
The documentation for this class was generated from the following file:

- bitmap_allocator.h

6.67 std::__detail::_BracketMatcher< _TraitsT, __icase, __collate > Struct Template Reference

`#include <regex_compiler.h>`

Public Types

- typedef _TraitsT::char_class_type **_CharClassT**
- typedef _TransT::_CharT **_CharT**
- typedef _TraitsT::string_type **_StringT**
- typedef _TransT::_StrTransT **_StrTransT**
- typedef _RegexTranslator< _TraitsT, __icase, __collate > **_TransT**

Public Member Functions

- **_BracketMatcher** (bool __is_non_matching, const _TraitsT &__traits)
- void **_M_add_char** (_CharT __c)
- void **_M_add_character_class** (const _StringT &__s, bool __neg)
- _StringT **_M_add_collate_element** (const _StringT &__s)
- void **_M_add_equivalence_class** (const _StringT &__s)
- void **_M_make_range** (_CharT __l, _CharT __r)
- void **_M_ready** ()
- bool **operator()** (_CharT __ch) const

6.67.1 Detailed Description

template<typename _TraitsT, bool __icase, bool __collate>
struct std::__detail::_BracketMatcher< _TraitsT, __icase, __collate >

Matches a character range (bracket expression).
The documentation for this struct was generated from the following files:

- regex_compiler.h
- regex_compiler.tcc

6.68 __gnu_cxx::_Caster< _ToType > Struct Template Reference

`#include <cast.h>`

Public Types

- typedef _ToType::element_type * **type**

6.68.1 Detailed Description

template<typename _ToType>
struct __gnu_cxx::_Caster< _ToType >

These functions are here to allow containers to support non standard pointer types. For normal pointers, these resolve to the use of the standard cast operation. For other types the functions will perform the appropriate cast to/from the custom pointer class so long as that class meets the following conditions: 1) has a typedef element_type which names tehe type it points to. 2) has a get() const method which returns element_type*. 3) has a constructor which can take one element_type* argument. This type supports the semantics of the pointer cast operators (below.)
The documentation for this struct was generated from the following file:

- cast.h

6.69 __gnu_cxx::_Char_types< _CharT > Struct Template Reference

`#include <char_traits.h>`

Public Types

- typedef unsigned long **int_type**
- typedef std::streamoff **off_type**
- typedef std::streampos **pos_type**
- typedef std::mbstate_t **state_type**

6.69.1 Detailed Description

template<typename _CharT>
struct __gnu_cxx::_Char_types< _CharT >

Mapping from character type to associated types.

Note

This is an implementation class for the generic version of char_traits. It defines int_type, off_type, pos_type, and state_type. By default these are unsigned long, streamoff, streampos, and mbstate_t. Users who need a different set of types, but who don't need to change the definitions of any function defined in char_traits, can specialize __gnu_cxx::_Char_types while leaving __gnu_cxx::char_traits alone.

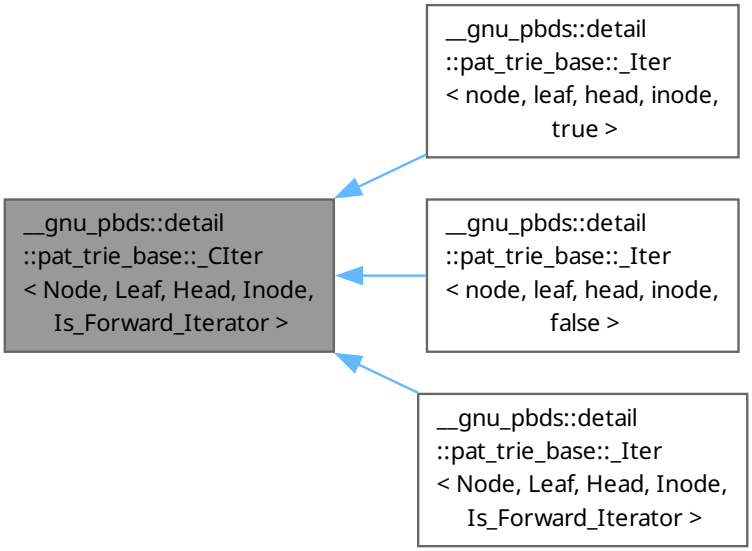
The documentation for this struct was generated from the following file:

- char_traits.h

6.70 __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > Class Template Reference

`#include <pat_trie_base.hpp>`

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >:



Public Types

- typedef allocator_type **_Alloc**
- typedef Node::allocator_type **allocator_type**
- typedef type_traits::const_pointer **const_pointer**
- typedef type_traits::const_reference **const_reference**
- typedef allocator_type::difference_type **difference_type**
- typedef rebind_traits< _Alloc, Head >::pointer **head_pointer**
- typedef Inode::iterator **inode_iterator**
- typedef rebind_traits< _Alloc, Inode >::pointer **inode_pointer**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef rebind_traits< _Alloc, Leaf >::const_pointer **leaf_const_pointer**
- typedef rebind_traits< _Alloc, Leaf >::pointer **leaf_pointer**
- typedef rebind_traits< _Alloc, Node >::pointer **node_pointer**
- typedef type_traits::pointer **pointer**
- typedef type_traits::reference **reference**
- typedef Node::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_CIter** (const _CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other)
- **_CIter** (node_pointer p_nd=0)
- bool **operator!=** (const _CIter &other) const
- bool **operator!=** (const _CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other) const
- const_reference **operator*** () const

- _CIter & **operator++** ()
- _CIter **operator++** (int)
- _CIter & **operator--** ()
- _CIter **operator--** (int)
- const_pointer **operator->** () const
- _CIter & **operator=** (const _CIter &other)
- _CIter & **operator=** (const _CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other)
- bool **operator==** (const _CIter &other) const
- bool **operator==** (const _CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other) const

Public Attributes

- node_pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

Static Protected Member Functions

- static node_pointer **get_larger_sibling** (node_pointer p_nd)
- static node_pointer **get_smaller_sibling** (node_pointer p_nd)
- static leaf_pointer **leftmost_descendant** (node_pointer p_nd)
- static leaf_pointer **rightmost_descendant** (node_pointer p_nd)

6.70.1 Detailed Description

template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >

Const iterator.
The documentation for this class was generated from the following file:

- pat_trie_base.hpp

6.71 std::__detail::_Compiler< _TraitsT > Class Template Reference

`#include <regex_compiler.h>`

Public Types

- typedef _TraitsT::char_type **_CharT**
- typedef regex_constants::syntax_option_type **_FlagT**
- typedef _NFA< _TraitsT > **_RegexT**

Public Member Functions

- **_Compiler** (const _CharT *__b, const _CharT *__e, const typename _TraitsT::locale_type &__traits, _FlagT __flags)
- shared_ptr< const _RegexT > **_M_get_nfa** () noexcept

6.71.1 Detailed Description

template<typename _TraitsT>
class std::__detail::_Compiler< _TraitsT >

Builds an NFA from an input iterator range.
The _TraitsT type should fulfill the requirements in C++11 28.3 [re.req].
The documentation for this class was generated from the following files:

- regex_compiler.h
- regex_compiler.tcc

6.72 std::__parallel::_CRandNumber< _MustBeInt > Struct Template Reference

`#include <algo.h>`

Public Member Functions

- int **operator()** (int __limit)

6.72.1 Detailed Description

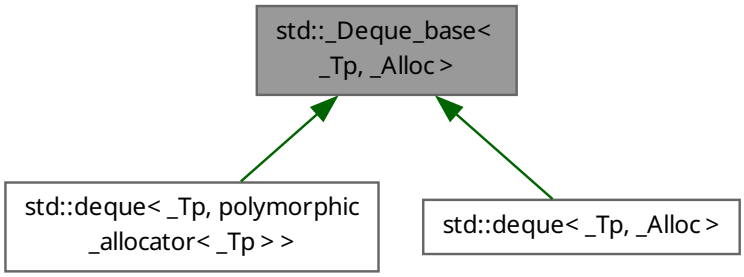
template<typename _MustBeInt = int>
struct std::__parallel::_CRandNumber< _MustBeInt >

Functor wrapper for std::rand().
The documentation for this struct was generated from the following file:

- algo.h

6.73 std::_Deque_base< _Tp, _Alloc > Class Template Reference

`#include <stl_deque.h>`
Inheritance diagram for std::_Deque_base< _Tp, _Alloc >:



Protected Types

- enum { **_S_initial_map_size** }
- typedef __gnu_cxx::__alloc_traits< _Tp_alloc_type > **_Alloc_traits**
- typedef __gnu_cxx::__alloc_traits< _Map_alloc_type > **_Map_alloc_traits**

- typedef _Alloc_traits::template rebind< _Ptr >::other **_Map_alloc_type**
- typedef iterator::_Map_pointer **_Map_pointer**
- typedef _Alloc_traits::pointer **_Ptr**
- typedef _Alloc_traits::const_pointer **_Ptr_const**
- typedef __gnu_cxx::__alloc_traits< _Alloc >::template rebind< _Tp >::other **_Tp_alloc_type**
- typedef _Alloc **allocator_type**
- typedef _Deque_iterator< _Tp, const _Tp &, _Ptr_const > **const_iterator**
- typedef _Deque_iterator< _Tp, _Tp &, _Ptr > **iterator**

Protected Member Functions

- **_Deque_base** (_Deque_base &&__x)
- **_Deque_base** (_Deque_base &&__x, const allocator_type &__a)
- **_Deque_base** (_Deque_base &&__x, const allocator_type &__a, size_t __n)
- **_Deque_base** (const allocator_type &__a)
- **_Deque_base** (const allocator_type &__a, size_t __num_elements)
- **_Deque_base** (size_t __num_elements)
- _Map_pointer **_M_allocate_map** (size_t __n)
- _Ptr **_M_allocate_node** ()
- void **_M_create_nodes** (_Map_pointer __nstart, _Map_pointer __nfinish)
- void **_M_deallocate_map** (_Map_pointer __p, size_t __n) noexcept
- void **_M_deallocate_node** (_Ptr __p) noexcept
- void **_M_destroy_nodes** (_Map_pointer __nstart, _Map_pointer __nfinish) noexcept
- _Map_alloc_type **_M_get_map_allocator** () const noexcept
- const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- void _M_initialize_map (size_t)
- allocator_type **get_allocator** () const noexcept

Protected Attributes

- _Deque_impl **_M_impl**

6.73.1 Detailed Description

template<typename _Tp, typename _Alloc>
class std::_Deque_base< _Tp, _Alloc >

Deque base class. This class provides the unified face for deque's allocation. This class's constructor and destructor allocate and deallocate (but do not initialize) storage. This makes exception safety easier.
Nothing in this class ever constructs or destroys an actual Tp element. (Deque handles that itself.) Only/All memory management is performed here.

6.73.2 Member Function Documentation

_M_initialize_map()

```
template<typename _Tp, typename _Alloc>
void std::_Deque_base< _Tp, _Alloc >::_M_initialize_map (
            size_t __num_elements)  [protected]
```

Layout storage.

Parameters

__num_elements	The count of T's for which to allocate space at first.

The initial underlying memory layout is a bit complicated...
References std::max().
The documentation for this class was generated from the following file:

- stl_deque.h

6.74 std::_Deque_iterator< _Tp, _Ref, _Ptr > Struct Template Reference

`#include <stl_deque.h>`

Public Types

- typedef __ptr_rebind< _Ptr, _Tp > **_Elt_pointer**
- typedef __ptr_rebind< _Ptr, _Elt_pointer > **_Map_pointer**
- typedef _Deque_iterator **_Self**
- typedef __iter< const _Tp > **const_iterator**
- typedef ptrdiff_t **difference_type**
- typedef __iter< _Tp > **iterator**
- typedef std::random_access_iterator_tag **iterator_category**
- typedef _Ptr **pointer**
- typedef _Ref **reference**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **_Deque_iterator** (_Elt_pointer __x, _Map_pointer __y) noexcept
- **_Deque_iterator** (const _Deque_iterator &__x) noexcept
- template<typename _Iter, typename = _Require<is_same<_Self, const_iterator>, is_same<_Iter, iterator>>>
  _Deque_iterator (const _Iter &__x) noexcept
- iterator **_M_const_cast** () const noexcept
- void _M_set_node (_Map_pointer __new_node) noexcept
- reference **operator*** () const noexcept
- _Self & **operator++** () noexcept
- _Self **operator++** (int) noexcept
- _Self & **operator+=** (difference_type __n) noexcept
- _Self & **operator--** () noexcept
- _Self **operator--** (int) noexcept
- _Self & **operator-=** (difference_type __n) noexcept
- pointer **operator->** () const noexcept
- _Deque_iterator & **operator=** (const _Deque_iterator &)=default
- reference **operator[ ]** (difference_type __n) const noexcept

Static Public Member Functions

- static size_t **_S_buffer_size** () noexcept
- template<typename _Fn>
  static _Self **_S_for_each_segment** (_Self __first, _Self __last, _Fn __func)

Public Attributes

- _Elt_pointer **_M_cur**
- _Elt_pointer **_M_first**
- _Elt_pointer **_M_last**
- _Map_pointer **_M_node**

Static Public Attributes

- static const bool **_S_enable_for_each_segment**

Friends

- _Self **operator+** (const _Self &__x, difference_type __n) noexcept
- _Self **operator+** (difference_type __n, const _Self &__x) noexcept
- template<typename _RefR, typename _PtrR>
  difference_type **operator-** (const _Self &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept
- difference_type **operator-** (const _Self &__x, const _Self &__y) noexcept
- _Self **operator-** (const _Self &__x, difference_type __n) noexcept
- strong_ordering **operator<=>** (const _Self &__x, const _Self &__y) noexcept
- template<typename _RefR, typename _PtrR>
  bool **operator==** (const _Self &__x, const _Deque_iterator< _Tp, _RefR, _PtrR > &__y) noexcept
- bool **operator==** (const _Self &__x, const _Self &__y) noexcept

6.74.1 Detailed Description

template<typename _Tp, typename _Ref, typename _Ptr>
struct std::_Deque_iterator< _Tp, _Ref, _Ptr >

A deque::iterator.
Quite a bit of intelligence here. Much of the functionality of deque is actually passed off to this class. A deque holds two of these internally, marking its valid range. Access to elements is done as offsets of either of those two, relying on operator overloading in this class.
All the functions are op overloads except for _M_set_node.

6.74.2 Member Function Documentation

_M_set_node()

```
template<typename _Tp, typename _Ref, typename _Ptr>
void std::_Deque_iterator< _Tp, _Ref, _Ptr >::_M_set_node (
            _Map_pointer __new_node)  [inline], [noexcept]
```

Prepares to traverse new_node. Sets everything except _M_cur, which should therefore be set by the caller immediately afterwards, based on _M_first and _M_last.
The documentation for this struct was generated from the following files:

- stl_algobase.h
- stl_deque.h

6.75 __gnu_parallel::_DRandomShufflingGlobalData< _RAIter > Struct Template Reference

```
#include <random_shuffle.h>
```

Public Types

- typedef _TraitsType::difference_type **_DifferenceType**
- typedef std::iterator_traits< _RAIter > **_TraitsType**
- typedef _TraitsType::value_type **_ValueType**

Public Member Functions

- _DRandomShufflingGlobalData (_RAIter &__source)

Public Attributes

- _ThreadIndex * _M_bin_proc
- _DifferenceType ** _M_dist
- int _M_num_bins
- int _M_num_bits
- _RAIter & _M_source
- _DifferenceType * _M_starts
- _ValueType ** _M_temporaries

6.75.1 Detailed Description

template<typename _RAIter>
struct __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >

Data known to every thread participating in __gnu_parallel::__parallel_random_shuffle().

6.75.2 Constructor & Destructor Documentation

_DRandomShufflingGlobalData()

```
template<typename _RAIter>
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_DRandomShufflingGlobalData (
            _RAIter & __source)  [inline]
```

Constructor.
References _M_source.

6.75.3 Member Data Documentation

_M_bin_proc

```
template<typename _RAIter>
_ThreadIndex* __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_bin_proc
```

Number of the thread that will further process the corresponding bin.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_dist

```
template<typename _RAIter>
_DifferenceType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_dist
```

Two-dimensional array to hold the thread-bin distribution.
Dimensions (_M_num_threads + 1) __x (_M_num_bins + 1).
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_num_bins

```
template<typename _RAIter>
int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bins
```

Number of bins to distribute to.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_num_bits

```
template<typename _RAIter>
int __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_num_bits
```

Number of bits needed to address the bins.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_source

```
template<typename _RAIter>
_RAIter& __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_source
```

Begin iterator of the __source.
Referenced by _DRandomShufflingGlobalData(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_starts

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_starts
```

Start indexes of the threads' __chunks.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_temporaries

```
template<typename _RAIter>
_ValueType** __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >::_M_temporaries
```

Temporary arrays for each thread.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().
The documentation for this struct was generated from the following file:

- random_shuffle.h

6.76 __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator > Struct Template Reference

```
#include <random_shuffle.h>
```

Public Attributes

- _BinIndex __bins_end
- _BinIndex _M_bins_begin
- int _M_num_threads
- _DRandomShufflingGlobalData< _RAIter > * _M_sd
- uint32_t _M_seed

6.76.1 Detailed Description

template<typename _RAIter, typename _RandomNumberGenerator>
struct __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >

Local data for a thread participating in __gnu_parallel::__parallel_random_shuffle().

6.76.2 Member Data Documentation

__bins_end

```
template<typename _RAIter, typename _RandomNumberGenerator>
_BinIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::__bins_end
```

End index for bins taken care of by this thread.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs().

_M_bins_begin

```
template<typename _RAIter, typename _RandomNumberGenerator>
_BinIndex __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_bins_begin
```

Begin index for bins taken care of by this thread.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_num_threads

```
template<typename _RAIter, typename _RandomNumberGenerator>
int __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_num_threads
```

Number of threads participating in total.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_sd

```
template<typename _RAIter, typename _RandomNumberGenerator>
_DRandomShufflingGlobalData<_RAIter>* __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumber↩
Generator >::_M_sd
```

Pointer to global data.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().

_M_seed

```
template<typename _RAIter, typename _RandomNumberGenerator>
uint32_t __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >::_M_seed
```

Random _M_seed for this thread.
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), and __gnu_parallel::__parallel_random_shuffle_drs_pu().
The documentation for this struct was generated from the following file:

- random_shuffle.h

6.77 __gnu_parallel::_DummyReduct Struct Reference

```
#include <for_each_selectors.h>
```

Public Member Functions

- bool **operator()** (bool, bool) const

6.77.1 Detailed Description

Reduction function doing nothing.
The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.78 __gnu_debug::_Equal_to< _Type > Class Template Reference

`#include <safe_sequence.h>`

Public Member Functions

- constexpr **_Equal_to** (const _Type &__v)
- bool **operator()** (const _Type &__x) const

6.78.1 Detailed Description

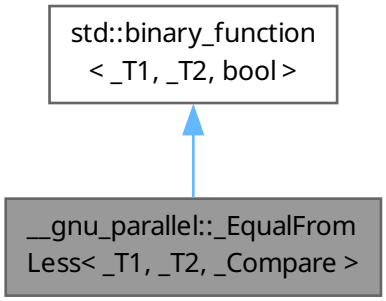
template<typename _Type>
class __gnu_debug::_Equal_to< _Type >

A simple function object that returns true if the passed-in value is equal to the stored value.
The documentation for this class was generated from the following file:

- safe_sequence.h

6.79 __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare > Class Template Reference

`#include <base.h>`
Inheritance diagram for __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >:



Public Types

- typedef _T1 first_argument_type
- typedef bool result_type
- typedef _T2 second_argument_type

Public Member Functions

- **_EqualFromLess** (_Compare &__comp)
- bool **operator()** (const _T1 &__a, const _T2 &__b)

6.79.1 Detailed Description

template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >

Constructs predicate for equality from strict weak ordering predicate.

6.79.2 Member Typedef Documentation

first_argument_type

```
typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< _T1, _T2, bool >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

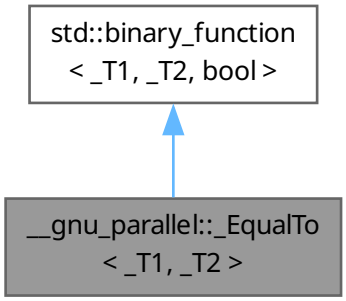
```
typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this class was generated from the following file:

- base.h

6.80 __gnu_parallel::_EqualTo< _T1, _T2 > Struct Template Reference

`#include <base.h>`
Inheritance diagram for __gnu_parallel::_EqualTo< _T1, _T2 >:



Public Types

- typedef _T1 first_argument_type
- typedef bool result_type
- typedef _T2 second_argument_type

Public Member Functions

- bool **operator()** (const _T1 &__t1, const _T2 &__t2) const

6.80.1 Detailed Description

template<typename _T1, typename _T2>
struct __gnu_parallel::_EqualTo< _T1, _T2 >

Similar to std::equal_to, but allows two different types.

6.80.2 Member Typedef Documentation

first_argument_type

`typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _T1, _T2, bool >::result_type [inherited]`
`result_type` is the return type

second_argument_type

`typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- base.h

6.81 std::__detail::_Executor< _BiIter, _Alloc, _TraitsT > Class Template Reference

`#include <regex_executor.h>`

Public Types

- typedef iterator_traits< _BiIter >::value_type **_CharT**
- typedef _TraitsT::char_class_type **_ClassT**
- typedef regex_constants::match_flag_type **_FlagT**
- typedef _NFA< _TraitsT > **_NFAT**
- typedef basic_regex< _CharT, _TraitsT > **_RegexT**
- typedef ::vector< sub_match< _BiIter >, _Alloc > **_ResultsVec**

Public Member Functions

- **_Executor** (_BiIter __begin, _BiIter __end, _ResultsVec &__results, const _RegexT &__re, _FlagT __flags, bool __use_dfs)
- bool **_M_match** ()
- bool **_M_search** ()
- bool **_M_search_from_first** ()

Public Attributes

- _Bilter **_M_begin**
- _ResultsVec **_M_cur_results**
- _Bilter **_M_current**
- const _Bilter **_M_end**
- _FlagT **_M_flags**
- ::vector< _ExecutorFrame< _Bilter > > **_M_frames**
- bool **_M_has_sol**
- ::vector< pair< _StateIdT, _ResultsVec > > **_M_match_queue**
- const _NFAT & **_M_nfa**
- const _RegexT & **_M_re**
- ::vector< pair< _Bilter, int > > **_M_rep_count**
- _ResultsVec & **_M_results**
- const _Search_mode **_M_search_mode**
- _Bilter **_M_sol_pos**
- _StateIdT **_M_start**
- bool * **_M_visited_states**

6.81.1 Detailed Description

template<typename _Bilter, typename _Alloc, typename _TraitsT>
class std::__detail::_Executor< _Bilter, _Alloc, _TraitsT >

Takes a regex and an input string and does the matching.
The _Executor class has two modes: DFS mode and BFS mode, controlled by the function parameter __search_mode.
The documentation for this class was generated from the following files:

- regex.h
- regex_executor.h
- regex_executor.tcc

6.82 __gnu_cxx::_ExtPtr_allocator< _Tp > Class Template Reference

`#include <extptr_allocator.h>`

Public Types

- typedef _Pointer_adapter< _Relative_pointer_impl< const _Tp > > **const_pointer**
- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Pointer_adapter< _Relative_pointer_impl< _Tp > > **pointer**
- typedef _Tp & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **_ExtPtr_allocator** (const _ExtPtr_allocator &__rarg) noexcept
- template<typename _Up>
  _ExtPtr_allocator (const _ExtPtr_allocator< _Up > &__rarg) noexcept
- const std::allocator< _Tp > & **_M_getUnderlyingImp** () const
- const_pointer **address** (const_reference __x) const noexcept
- pointer **address** (reference __x) const noexcept

- pointer **allocate** (size_type __n, const void *=0)
- template<typename _Up, typename... _Args>
  void **construct** (_Up *__p, _Args &&... __args)
- template<typename... _Args>
  void **construct** (pointer __p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up>
  void **destroy** (_Up *__p)
- void **destroy** (pointer __p)
- size_type **max_size** () const noexcept
- bool **operator==** (const _ExtPtr_allocator &__rarg) const
- template<typename _Up>
  bool **operator==** (const _ExtPtr_allocator< _Up > &__rarg) const

Friends

- template<typename _Up>
  void **swap** (_ExtPtr_allocator< _Up > &, _ExtPtr_allocator< _Up > &)

6.82.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::_ExtPtr_allocator< _Tp >

An example allocator which uses a non-standard pointer type.
This allocator specifies that containers use a 'relative pointer' as it's pointer type. (See ext/pointer.h) Memory allocation in this example is still performed using std::allocator.
The documentation for this class was generated from the following file:

- extptr_allocator.h

6.83 __gnu_cxx::__detail::_Ffit_finder< _Tp > Class Template Reference

`#include <bitmap_allocator.h>`

Public Types

- typedef _Block_pair **argument_type**
- typedef bool **result_type**

Public Member Functions

- std::size_t * **_M_get** () const throw ()
- _Counter_type **_M_offset** () const throw ()
- bool **operator()** (_Block_pair __bp) throw ()

6.83.1 Detailed Description

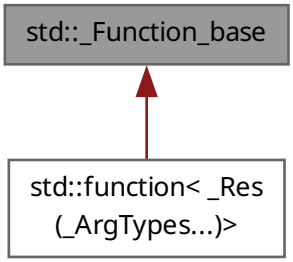
template<typename _Tp>
class __gnu_cxx::__detail::_Ffit_finder< _Tp >

The class which acts as a predicate for applying the first-fit memory allocation policy for the bitmap allocator.
The documentation for this class was generated from the following file:

- bitmap_allocator.h

6.84 std::_Function_base Class Reference

`#include <std_function.h>`
Inheritance diagram for std::_Function_base:



Public Types

- using **_Manager_type**

Public Member Functions

- bool **_M_empty** () const

Public Attributes

- _Any_data **_M_functor**
- _Manager_type **_M_manager**

Static Public Attributes

- static const size_t **_M_max_align**
- static const size_t **_M_max_size**

6.84.1 Detailed Description

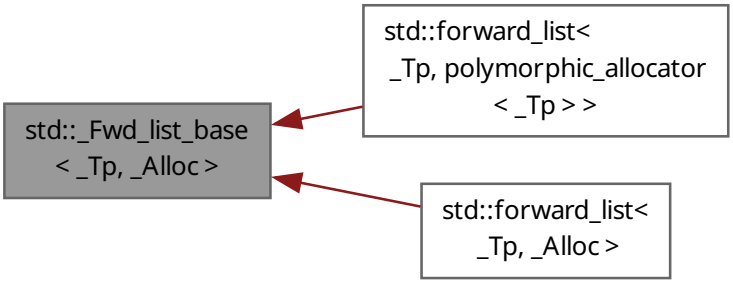
Base class of all polymorphic function object wrappers.
The documentation for this class was generated from the following file:

- std_function.h

6.85 std::_Fwd_list_base< _Tp, _Alloc > Struct Template Reference

`#include <forward_list.h>`

Inheritance diagram for std::_Fwd_list_base< _Tp, _Alloc >:



Public Types

- using **const_iterator**
- using **iterator**
- using **pointer**

Public Member Functions

- **_Fwd_list_base** (_Fwd_list_base &&)=default
- **_Fwd_list_base** (_Fwd_list_base &&__lst, _Node_alloc_type &&__a)
- **_Fwd_list_base** (_Fwd_list_base &&__lst, _Node_alloc_type &&__a, std::true_type)
- **_Fwd_list_base** (_Node_alloc_type &&__a)
- const _Node_alloc_type & **_M_get_Node_allocator** () const noexcept
- _Node_alloc_type & **_M_get_Node_allocator** () noexcept

Protected Types

- using **_Base_ptr**
- using **_Node**
- using **_Node_alloc_traits**
- using **_Node_alloc_type**
- using **_Node_ptr**
- using **_Node_traits**

Protected Member Functions

- template<typename... _Args>
  _Node_ptr **_M_create_node** (_Args &&... __args)
- void **_M_destroy_node** (_Node_ptr __p)
- _Base_ptr **_M_erase_after** (_Base_ptr __pos)
- _Base_ptr **_M_erase_after** (_Base_ptr __pos, _Base_ptr __last)
- _Node * **_M_get_node** ()
- template<typename... _Args>
  _Base_ptr **_M_insert_after** (const_iterator __pos, _Args &&... __args)
- void **_M_put_node** (_Node_ptr __p)

Protected Attributes

- _Fwd_list_impl **_M_impl**

6.85.1 Detailed Description

template<typename _Tp, typename _Alloc>
struct std::_Fwd_list_base< _Tp, _Alloc >

Base class for forward_list.
The documentation for this struct was generated from the following files:

- forward_list.h
- forward_list.tcc

6.86 std::_Fwd_list_const_iterator< _Tp > Struct Template Reference

`#include <forward_list.h>`

Public Types

- typedef const _Fwd_list_node< _Tp > **_Node**
- typedef _Fwd_list_const_iterator< _Tp > **_Self**
- typedef ptrdiff_t **difference_type**
- typedef _Fwd_list_iterator< _Tp > **iterator**
- typedef std::forward_iterator_tag **iterator_category**
- typedef const _Tp * **pointer**
- typedef const _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- **_Fwd_list_const_iterator** (const _Fwd_list_node_base *__n) noexcept
- **_Fwd_list_const_iterator** (const iterator &__iter) noexcept
- reference **operator*** () const noexcept
- _Self & **operator++** () noexcept
- _Self **operator++** (int) noexcept
- pointer **operator->** () const noexcept

Friends

- template<typename, typename>
  struct **_Fwd_list_base**
- template<typename, typename>
  class **forward_list**
- bool operator== (const _Self &__x, const _Self &__y) noexcept

6.86.1 Detailed Description

template<typename _Tp>
struct std::_Fwd_list_const_iterator< _Tp >

A forward_list::const_iterator.
All the functions are op overloads.

6.86.2 Friends And Related Symbol Documentation

operator==

```
template<typename _Tp>
bool operator== (
            const _Self & __x,
            const _Self & __y)  [friend]
```

Forward list const_iterator equality comparison.
The documentation for this struct was generated from the following file:

- forward_list.h

6.87 std::_Fwd_list_iterator< _Tp > Struct Template Reference

`#include <forward_list.h>`

Public Types

- typedef _Fwd_list_node< _Tp > **_Node**
- typedef _Fwd_list_iterator< _Tp > **_Self**
- typedef ptrdiff_t **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- **_Fwd_list_iterator** (_Fwd_list_node_base *__n) noexcept
- reference **operator*** () const noexcept
- _Self & **operator++** () noexcept
- _Self **operator++** (int) noexcept
- pointer **operator->** () const noexcept

Friends

- template<typename, typename>
  struct **_Fwd_list_base**
- struct **_Fwd_list_const_iterator< _Tp >**
- template<typename, typename>
  class **forward_list**
- bool operator== (const _Self &__x, const _Self &__y) noexcept

6.87.1 Detailed Description

template<typename _Tp>
struct std::_Fwd_list_iterator< _Tp >

A forward_list::iterator.
All the functions are op overloads.

6.87.2 Friends And Related Symbol Documentation

operator==

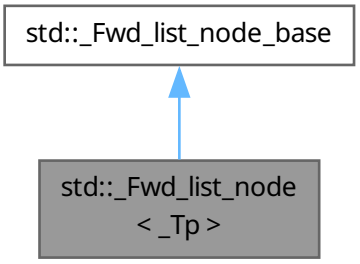
```
template<typename _Tp>
bool operator== (
            const _Self & __x,
            const _Self & __y)  [friend]
```

Forward list iterator equality comparison.
The documentation for this struct was generated from the following file:

- forward_list.h

6.88 std::_Fwd_list_node< _Tp > Struct Template Reference

`#include <forward_list.h>`
Inheritance diagram for std::_Fwd_list_node< _Tp >:



Public Types

- using **_Base_ptr**
- using **_Node_ptr**

Public Member Functions

- _Fwd_list_node_base * **_M_base_ptr** ()
- const _Fwd_list_node_base * **_M_base_ptr** () const
- _Node_ptr **_M_node_ptr** ()
- void **_M_reverse_after** () noexcept
- _Fwd_list_node_base * **_M_transfer_after** (_Fwd_list_node_base *__begin, _Fwd_list_node_base *__end) noexcept
- const _Tp * **_M_valptr** () const noexcept
- _Tp * **_M_valptr** () noexcept

Public Attributes

- _Fwd_list_node_base * **_M_next**
- __gnu_cxx::__aligned_buffer< _Tp > **_M_storage**

6.88.1 Detailed Description

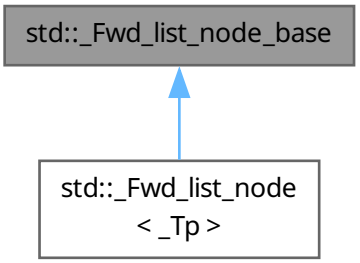
template<**typename _Tp**>
struct std::_Fwd_list_node< **_Tp** >

A helper node class for forward_list. This is just a linked list with uninitialized storage for a data value in each node. There is a sorting utility method.
The documentation for this struct was generated from the following file:

- forward_list.h

6.89 std::_Fwd_list_node_base Struct Reference

`#include <forward_list.h>`
Inheritance diagram for std::_Fwd_list_node_base:



Public Types

- using **_Base_ptr**

Public Member Functions

- **_Fwd_list_node_base** (_Fwd_list_node_base &&__x) noexcept
- **_Fwd_list_node_base** (const _Fwd_list_node_base &)=delete
- _Fwd_list_node_base * **_M_base_ptr** ()
- const _Fwd_list_node_base * **_M_base_ptr** () const
- void **_M_reverse_after** () noexcept
- _Fwd_list_node_base * **_M_transfer_after** (_Fwd_list_node_base *__begin, _Fwd_list_node_base *__end) noexcept
- _Fwd_list_node_base & **operator=** (_Fwd_list_node_base &&__x) noexcept
- _Fwd_list_node_base & **operator=** (const _Fwd_list_node_base &)=delete

Public Attributes

- _Fwd_list_node_base * **_M_next**

6.89.1 Detailed Description

A helper basic node class for forward_list.

```
This is just a linked list with nothing inside it.
There are purely list shuffling utility methods here.
```

The documentation for this struct was generated from the following file:

- forward_list.h

6.90 __gnu_parallel::_GuardedIterator< _RAIter, _Compare > Class Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- _GuardedIterator (_RAIter __begin, _RAIter __end, _Compare &__comp)
- operator _RAIter () const
- std::iterator_traits< _RAIter >::value_type & operator* () const
- _GuardedIterator< _RAIter, _Compare > & operator++ ()

Friends

- bool operator< (const _GuardedIterator< _RAIter, _Compare > &__bi1, const _GuardedIterator< _RAIter, _Compare > &__bi2)
- bool operator<= (const _GuardedIterator< _RAIter, _Compare > &__bi1, const _GuardedIterator< _RAIter, _Compare > &__bi2)

6.90.1 Detailed Description

template<typename _RAIter, typename _Compare>
class __gnu_parallel::_GuardedIterator< _RAIter, _Compare >

_Iterator wrapper supporting an implicit supremum at the end of the sequence, dominating all comparisons.
The implicit supremum comes with a performance cost.
Deriving from _RAIter is not possible since _RAIter need not be a class.

6.90.2 Constructor & Destructor Documentation

_GuardedIterator()

```
template<typename _RAIter, typename _Compare>
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >::_GuardedIterator (
            _RAIter __begin,
            _RAIter __end,
            _Compare & __comp)  [inline]
```

Constructor. Sets iterator to beginning of sequence.

Parameters

__begin	Begin iterator of sequence.
__end	End iterator of sequence.
__comp	Comparator provided for associated overloaded compare operators.

Referenced by operator<, and operator<=.

6.90.3 Member Function Documentation

operator _RAIter()

```
template<typename _RAIter, typename _Compare>
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >::operator _RAIter () const  [inline]
```

Convert to wrapped iterator.

Returns

Wrapped iterator.

operator*()

```
template<typename _RAIter, typename _Compare>
std::iterator_traits< _RAIter >::value_type & __gnu_parallel::_GuardedIterator< _RAIter, _Compare
>::operator* () const  [inline]
```

Dereference operator.

Returns

Referenced element.

operator++()

```
template<typename _RAIter, typename _Compare>
_GuardedIterator< _RAIter, _Compare > & __gnu_parallel::_GuardedIterator< _RAIter, _Compare >::↩
operator++ ()  [inline]
```

Pre-increment operator.

Returns

This.

6.90.4 Friends And Related Symbol Documentation

operator<

```
template<typename _RAIter, typename _Compare>
bool operator< (
            const _GuardedIterator< _RAIter, _Compare > & __bi1,
            const _GuardedIterator< _RAIter, _Compare > & __bi2)  [friend]
```

Compare two elements referenced by guarded iterators.

Parameters

__bi1	First iterator.
__bi2	Second iterator.

Returns

`true` if less.

References _GuardedIterator().

operator<=

```
template<typename _RAIter, typename _Compare>
bool operator<= (
            const _GuardedIterator< _RAIter, _Compare > & __bi1,
            const _GuardedIterator< _RAIter, _Compare > & __bi2)  [friend]
```

Compare two elements referenced by guarded iterators.

Parameters

__bi1	First iterator.
__bi2	Second iterator.

Returns

`True` if less equal.

References _GuardedIterator().

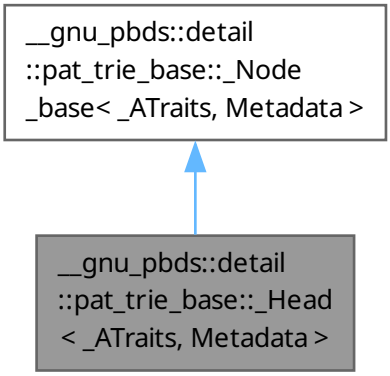
The documentation for this class was generated from the following file:

- multiway_merge.h

6.91 __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata > Struct Template Reference

`#include <pat_trie_base.hpp>`

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >:



Public Types

- typedef _ATraits::const_iterator **a_const_iterator**
- typedef detail::rebind_traits< _Alloc, _ATraits >::const_pointer **a_const_pointer**
- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**

- typedef _Node_base< _ATraits, Metadata > **base_type**
- typedef base_type::node_pointer **node_pointer**
- typedef base_type::type_traits **type_traits**

Public Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_min**
- node_pointer **m_p_parent**
- const node_type **m_type**

6.91.1 Detailed Description

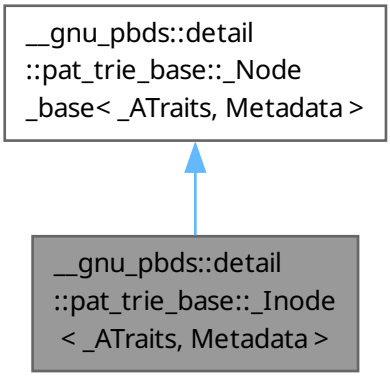
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >

Head node for PATRICIA tree.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

6.92 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata > Struct Template Reference

`#include <pat_trie_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >:



Classes

- struct const_iterator
- struct iterator

Public Types

- enum { **arr_size** }
- typedef detail::rebind_traits< _Alloc, node_pointer > **__rebind_np**
- typedef base_type::allocator_type **_Alloc**
- typedef base_type::access_traits **access_traits**
- typedef _Alloc **allocator_type**
- typedef _Node_base< _ATraits, Metadata > **base_type**
- typedef __rebind_np::pointer **node_pointer_pointer**
- typedef __rebind_np::reference **node_pointer_reference**
- typedef _Alloc::size_type **size_type**
- typedef base_type::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_Inode** (size_type, const a_const_iterator)
- node_pointer **add_child** (node_pointer, a_const_iterator, a_const_iterator, a_const_pointer)
- iterator **begin** ()
- const_iterator **begin** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **get_child_it** (a_const_iterator, a_const_iterator, a_const_pointer)
- node_pointer **get_child_node** (a_const_iterator, a_const_iterator, a_const_pointer)
- node_const_pointer **get_child_node** (a_const_iterator, a_const_iterator, a_const_pointer) const
- size_type **get_e_ind** () const
- node_const_pointer **get_join_child** (node_const_pointer, a_const_pointer) const
- node_pointer **get_join_child** (node_pointer, a_const_pointer)
- node_pointer **get_lower_bound_child_node** (a_const_iterator, a_const_iterator, size_type, a_const_pointer)
- leaf_pointer **leftmost_descendant** ()
- leaf_const_pointer **leftmost_descendant** () const
- a_const_iterator **pref_b_it** () const
- a_const_iterator **pref_e_it** () const
- void **remove_child** (iterator)
- void **remove_child** (node_pointer)
- void **replace_child** (node_pointer, a_const_iterator, a_const_iterator, a_const_pointer)
- leaf_pointer **rightmost_descendant** ()
- leaf_const_pointer **rightmost_descendant** () const
- bool **should_be_mine** (a_const_iterator, a_const_iterator, size_type, a_const_pointer) const
- void **update_prefixes** (a_const_pointer)

Public Attributes

- node_pointer **m_p_parent**
- const node_type **m_type**

6.92.1 Detailed Description

template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >

Internal node type, PATRICIA tree.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

6.93 __gnu_cxx::_Invalid_type Struct Reference

`#include <pointer.h>`

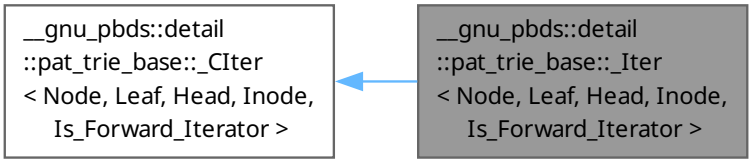
6.93.1 Detailed Description

The specialization on this type helps resolve the problem of reference to void, and eliminates the need to specialize _Pointer_adapter for cases of void∗, const void∗, and so on.
The documentation for this struct was generated from the following file:

- pointer.h

6.94 __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator > Class Template Reference

`#include <pat_trie_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >:



Public Types

- typedef allocator_type **_Alloc**
- typedef base_type::allocator_type **allocator_type**
- typedef _CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator > **base_type**
- typedef type_traits::const_pointer **const_pointer**
- typedef type_traits::const_reference **const_reference**
- typedef allocator_type::difference_type **difference_type**
- typedef base_type::head_pointer **head_pointer**
- typedef Inode::iterator **inode_iterator**
- typedef base_type::inode_pointer **inode_pointer**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef base_type::leaf_const_pointer **leaf_const_pointer**
- typedef base_type::leaf_pointer **leaf_pointer**
- typedef base_type::node_pointer **node_pointer**
- typedef type_traits::pointer **pointer**
- typedef type_traits::reference **reference**
- typedef base_type::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_Iter** (const _Iter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other)
- **_Iter** (node_pointer p_nd=0)
- bool **operator!=** (const _CIter &other) const
- bool **operator!=** (const _CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other) const
- reference **operator*** () const
- _Iter & **operator++** ()
- _Iter **operator++** (int)
- _Iter & **operator--** ()
- _Iter **operator--** (int)
- pointer **operator->** () const
- _Iter & **operator=** (const _Iter &other)
- _Iter & **operator=** (const _Iter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other)
- bool **operator==** (const _CIter &other) const
- bool **operator==** (const _CIter< Node, Leaf, Head, Inode, !Is_Forward_Iterator > &other) const

Public Attributes

- node_pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

Static Protected Member Functions

- static node_pointer **get_larger_sibling** (node_pointer p_nd)
- static node_pointer **get_smaller_sibling** (node_pointer p_nd)
- static leaf_pointer **leftmost_descendant** (node_pointer p_nd)
- static leaf_pointer **rightmost_descendant** (node_pointer p_nd)

6.94.1 Detailed Description

template<typename Node, typename Leaf, typename Head, typename Inode, bool Is_Forward_Iterator>
class __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >

Iterator.
The documentation for this class was generated from the following file:

- pat_trie_base.hpp

6.95 std::__fwdlist::_Iterator< _Const, _Ptr > Class Template Reference

`#include <forward_list.h>`

Public Types

- using **difference_type**
- using **iterator_category**
- using **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- constexpr **_Iterator** (_Base_ptr __x) noexcept
- **_Iterator** (const _Iterator &)=default
- constexpr **_Iterator** (const _Iterator< false, _Ptr > &__i)
- constexpr reference **operator*** () const noexcept
- constexpr _Iterator & **operator++** () noexcept
- constexpr _Iterator **operator++** (int) noexcept
- constexpr pointer **operator->** () const noexcept
- _Iterator & **operator=** (const _Iterator &)=default

Friends

- template<typename _Tp, typename _Allocator>
  struct **::_Fwd_list_base**
- template<typename _Tp, typename _Allocator>
  class **::forward_list**
- constexpr bool operator== (const _Iterator &__x, const _Iterator &__y) noexcept

6.95.1 Detailed Description

template<bool _Const, typename _Ptr>
class std::__fwdlist::_Iterator< _Const, _Ptr >

A forward_list iterator when the allocator uses fancy pointers.

6.95.2 Friends And Related Symbol Documentation

operator==

```
template<bool _Const, typename _Ptr>
bool operator== (
            const _Iterator< _Const, _Ptr > & __x,
            const _Iterator< _Const, _Ptr > & __y)  [friend]
```

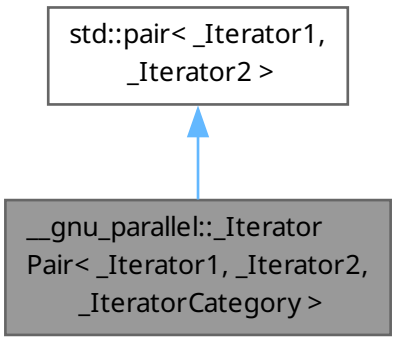
Forward list iterator equality comparison.
The documentation for this class was generated from the following file:

- forward_list.h

6.96 __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory > Class Template Reference

```
#include <iterator.h>
```

Inheritance diagram for __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >:



Public Types

- typedef std::iterator_traits< _Iterator1 > **_TraitsType**
- typedef _TraitsType::difference_type **difference_type**
- typedef _Iterator1 first_type
- typedef _IteratorCategory **iterator_category**
- typedef _IteratorPair * **pointer**
- typedef _IteratorPair & **reference**
- typedef _Iterator2 second_type
- typedef void **value_type**

Public Member Functions

- **_IteratorPair** (const _Iterator1 &__first, const _Iterator2 &__second)
- constexpr **explicit** (!_S_convertible< _U1 &, _U2 & >()) pair(pair< _U1
- constexpr **explicit** (!_S_convertible< _U1 &, _U2 & >()) pair(pair< _U1
- constexpr _U2 &&constexpr explicit (!_S_convertible< _U1 &, _U2 & >()) pair(pair< _U1
- constexpr _U2 &&constexpr explicit (!_S_convertible< _U1 &, _U2 & >()) pair(pair< _U1
- constexpr **explicit** (!_S_convertible< _U1, _U2 >()) pair(_U1 &&
- constexpr **explicit** (!_S_convertible< _U1, _U2 >()) pair(_U1 &&
- constexpr explicit (!_S_convertible< _U1, _U2 >()) pair(_U1 &&__x
- constexpr explicit (!_S_convertible< _U1, _U2 >()) pair(_U1 &&__x
- constexpr **explicit** (!_S_convertible< _U1, _U2 >()) pair(pair< _U1
- constexpr **explicit** (!_S_convertible< _U1, _U2 >()) pair(pair< _U1
- constexpr _U2 &constexpr explicit (!_S_convertible< _U1, _U2 >()) pair(pair< _U1
- constexpr _U2 &constexpr explicit (!_S_convertible< _U1, _U2 >()) pair(pair< _U1
- constexpr explicit (!_S_convertible< const _Iterator1 &, const _Iterator2 & >()) pair(const type_identity_t< _↩ Iterator1 > &__x
- constexpr explicit (!_S_convertible< const _Iterator1 &, const _Iterator2 & >()) pair(const type_identity_t< _↩ Iterator1 > &__x
- constexpr **explicit** (!_S_convertible< const _U1 &, const _U2 & >()) pair(const pair< _U1
- constexpr **explicit** (!_S_convertible< const _U1 &, const _U2 & >()) pair(const pair< _U1

- constexpr _U2 &&constexpr explicit (!_S_convertible< const _U1 &, const _U2 & >()) pair(const pair< _U1
- constexpr _U2 &&constexpr explicit (!_S_convertible< const _U1 &, const _U2 & >()) pair(const pair< _U1
- constexpr **explicit** (!_S_convertible< const _U1, const _U2 >()) pair(const pair< _U1
- constexpr **explicit** (!_S_convertible< const _U1, const _U2 >()) pair(const pair< _U1
- constexpr _U2 &constexpr explicit (!_S_convertible< const _U1, const _U2 >()) pair(const pair< _U1
- constexpr _U2 &constexpr explicit (!_S_convertible< const _U1, const _U2 >()) pair(const pair< _U1
- constexpr **explicit** (!_S_convertible_from_pair_like< _UPair >()) pair(_UPair &&)=delete
- constexpr **explicit** (!_S_convertible_from_pair_like< _UPair >()) pair(_UPair &&)=delete
- constexpr _U2 &&constexpr **explicit** (!_S_convertible_from_pair_like< _UPair >()) pair(_UPair &&__p)
- constexpr _U2 &&constexpr **explicit** (!_S_convertible_from_pair_like< _UPair >()) pair(_UPair &&__p)
- constexpr explicit (__not_< __and_< __is_implicitly_default_constructible< _Iterator1 >, __is_implicitly_default_constructible< _Iterator2 > > >()) pair() noexcept(is_nothrow_default_constructible_v< _Iterator1 > &&is_nothrow_default_constructible_v< _Iterator2 >)
- constexpr explicit (__not_< __and_< __is_implicitly_default_constructible< _Iterator1 >, __is_implicitly_default_constructible< _Iterator2 > > >()) pair() noexcept(is_nothrow_default_constructible_v< _Iterator1 > &&is_nothrow_default_constructible_v< _Iterator2 >)
- constexpr _U2 &&constexpr _U2 &__p **noexcept** (_S_nothrow_constructible< _U1 &, _U2 & >())
- constexpr _U2 &&constexpr _U2 &__p **noexcept** (_S_nothrow_constructible< _U1 &, _U2 & >())
- constexpr _U2 &constexpr _U2 &&__p **noexcept** (_S_nothrow_constructible< _U1, _U2 >())
- constexpr _U2 &constexpr _U2 &&__p **noexcept** (_S_nothrow_constructible< _U1, _U2 >())
- constexpr _U2 &&__y **noexcept** (_S_nothrow_constructible< _U1, _U2 >())
- constexpr _U2 &&__y **noexcept** (_S_nothrow_constructible< _U1, _U2 >())
- constexpr const _Iterator2 &__y **noexcept** (_S_nothrow_constructible< const _Iterator1 &, const _Iterator2 & >())
- constexpr const _Iterator2 &__y **noexcept** (_S_nothrow_constructible< const _Iterator1 &, const _Iterator2 & >())
- constexpr _U2 &&constexpr _U2 &__p **noexcept** (_S_nothrow_constructible< const _U1 &, const _U2 & >())
- constexpr _U2 &&constexpr _U2 &__p **noexcept** (_S_nothrow_constructible< const _U1 &, const _U2 & >())
- constexpr _U2 &constexpr _U2 &&__p **noexcept** (_S_nothrow_constructible< const _U1, const _U2 >())
- constexpr _U2 &constexpr _U2 &&__p **noexcept** (_S_nothrow_constructible< const _U1, const _U2 >())
- **operator _Iterator2** () const
- _IteratorPair **operator+** (difference_type __delta) const
- _IteratorPair & **operator++** ()
- const _IteratorPair **operator++** (int)
- difference_type **operator-** (const _IteratorPair &__other) const
- _IteratorPair & **operator--** ()
- const _IteratorPair **operator--** (int)
- _IteratorPair & **operator=** (const _IteratorPair &__other)
- constexpr void **swap** (const pair &__p) const noexcept(__and_v< __is_nothrow_swappable< const _Iterator1 >, __is_nothrow_swappable< const _Iterator2 > >)
- constexpr void **swap** (const pair &__p) const noexcept(__and_v< __is_nothrow_swappable< const _Iterator1 >, __is_nothrow_swappable< const _Iterator2 > >)
- constexpr void swap (pair &__p) noexcept(__and_< __is_nothrow_swappable< _Iterator1 >, __is_nothrow_swappable< _Iterator2 > >::value)
- constexpr void swap (pair &__p) noexcept(__and_< __is_nothrow_swappable< _Iterator1 >, __is_nothrow_swappable< _Iterator2 > >::value)

Public Attributes

- _Iterator1 first
- _Iterator1 first
- _Iterator2 second
- _Iterator2 second

Related Symbols

(Note that these are not member symbols.)

- constexpr enable_if< __and_< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type swap (pair< _Iterator1, _Iterator2 > &__x, pair< _Iterator1, _Iterator2 > &__y) noexcept(noexcept(__x.swap(↩ __y)))
- constexpr enable_if< __and_< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type swap (pair< _Iterator1, _Iterator2 > &__x, pair< _Iterator1, _Iterator2 > &__y) noexcept(noexcept(__x.swap(↩ __y)))
- constexpr bool operator== (const pair< _Iterator1, _Iterator2 > &__x, const pair< _U1, _U2 > &__y)
- constexpr bool operator== (const pair< _Iterator1, _Iterator2 > &__x, const pair< _U1, _U2 > &__y)
- constexpr common_comparison_category_t< __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__↩ synth3way_t< _Iterator2, _U2 > > operator<=> (const pair< _Iterator1, _Iterator2 > &__x, const pair< _U1, _U2 > &__y)
- constexpr common_comparison_category_t< __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__↩ synth3way_t< _Iterator2, _U2 > > operator<=> (const pair< _Iterator1, _Iterator2 > &__x, const pair< _U1, _U2 > &__y)

6.96.1 Detailed Description

template<typename _Iterator1, typename _Iterator2, typename _IteratorCategory>
class __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >

A pair of iterators. The usual iterator operations are applied to both child iterators.

6.96.2 Member Typedef Documentation

first_type

```
typedef _Iterator1 std::pair< _Iterator1, _Iterator2 >::first_type  [inherited]
```

The type of the `first` member.

second_type

```
typedef _Iterator2 std::pair< _Iterator1, _Iterator2 >::second_type  [inherited]
```

The type of the `second` member.

6.96.3 Member Function Documentation

explicit() [1/14]

```
_U2 &&constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
            !_S_convertible< _U1 &, _U2 & > ())  [constexpr], [inherited]
```

Converting constructor from a non-const `pair<U1, U2>` lvalue.

explicit() [2/14]

```
_U2 &&constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
            !_S_convertible< _U1 &, _U2 & > ())  [constexpr], [inherited]
```

Converting constructor from a non-const `pair<U1, U2>` lvalue.

explicit() [3/14]

```
std::pair< _Iterator1, _Iterator2 >::explicit (
            !_S_convertible< _U1, _U2 > ()) &&  [constexpr], [inherited]
```

Constructor accepting two values of arbitrary types.

explicit() [4/14]

```
std::pair< _Iterator1, _Iterator2 >::explicit (
            !_S_convertible< _U1, _U2 > ()) &&  [constexpr], [inherited]
```

Constructor accepting two values of arbitrary types.

explicit() [5/14]

```
_U2 &constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
            !_S_convertible< _U1, _U2 > ())  [constexpr], [inherited]
```

Converting constructor from a non-const `pair<U1, U2>` rvalue.

explicit() [6/14]

```
_U2 &constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
            !_S_convertible< _U1, _U2 > ())  [constexpr], [inherited]
```

Converting constructor from a non-const `pair<U1, U2>` rvalue.

explicit() [7/14]

```
std::pair< _Iterator1, _Iterator2 >::explicit (
            !_S_convertible< const _Iterator1 &, const _Iterator2 & > ()) const &  [constexpr],
[inherited]
```

Constructor accepting lvalues of `first_type` and `second_type`.

explicit() [8/14]

```
std::pair< _Iterator1, _Iterator2 >::explicit (
            !_S_convertible< const _Iterator1 &, const _Iterator2 & > ()) const &  [constexpr],
[inherited]
```

Constructor accepting lvalues of `first_type` and `second_type`.

explicit() [9/14]

```
_U2 &&constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
            !_S_convertible< const _U1 &, const _U2 & > ()) const  [constexpr], [inherited]
```

Converting constructor from a const `pair<U1, U2>` lvalue.

explicit() [10/14]

```
_U2 &&constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
            !_S_convertible< const _U1 &, const _U2 & > ()) const  [constexpr], [inherited]
```

Converting constructor from a const `pair<U1, U2>` lvalue.

explicit() [11/14]

```
_U2 &constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
            !_S_convertible< const _U1, const _U2 > ()) const  [constexpr], [inherited]
```

Converting constructor from a const `pair<U1, U2>` rvalue.

explicit() [12/14]

```
_U2 &constexpr std::pair< _Iterator1, _Iterator2 >::explicit (
            !_S_convertible< const _U1, const _U2 > ()) const  [constexpr], [inherited]
```

Converting constructor from a const `pair<U1, U2>` rvalue.

explicit() [13/14]

```
std::pair< _Iterator1, _Iterator2 >::explicit (
            __not_< __and_< __is_implicitly_default_constructible< _Iterator1 >, __is_implicitly↩
_default_constructible< _Iterator2 > > > ()) [inline], [constexpr], [noexcept], [inherited]
```

Default constructor.

explicit() [14/14]

```
std::pair< _Iterator1, _Iterator2 >::explicit (
            __not_< __and_< __is_implicitly_default_constructible< _Iterator1 >, __is_implicitly↩
_default_constructible< _Iterator2 > > > ()) [inline], [constexpr], [noexcept], [inherited]
```

Default constructor.

swap() [1/2]

```
void std::pair< _Iterator1, _Iterator2 >::swap (
            pair< _Iterator1, _Iterator2 > & __p) [inline], [constexpr], [noexcept], [inherited]
```

Swap the first members and then the second members.

swap() [2/2]

```
void std::pair< _Iterator1, _Iterator2 >::swap (
            pair< _Iterator1, _Iterator2 > & __p) [inline], [constexpr], [noexcept], [inherited]
```

Swap the first members and then the second members.

6.96.4 Friends And Related Symbol Documentation

operator<=>() [1/2]

```
common_comparison_category_t< __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__synth3way↩
_t< _Iterator2, _U2 > > operator<=> (
            const pair< _Iterator1, _Iterator2 > & __x,
            const pair< _U1, _U2 > & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

operator<=>() [2/2]

```
common_comparison_category_t< __detail::__synth3way_t< _Iterator1, _U1 >, __detail::__synth3way↩
_t< _Iterator2, _U2 > > operator<=> (
            const pair< _Iterator1, _Iterator2 > & __x,
            const pair< _U1, _U2 > & __y) [related]
```

Defines a lexicographical order for pairs.

For two pairs of comparable types, `P` is ordered before `Q` if `P.first` is less than `Q.first`, or if `P.first` and `Q.first` are equivalent (neither is less than the other) and `P.second` is less than `Q.second`.

operator==() [1/2]

```
bool operator== (
            const pair< _Iterator1, _Iterator2 > & __x,
            const pair< _U1, _U2 > & __y) [related]
```

Two pairs are equal iff their members are equal.

operator==() [2/2]

```
bool operator== (
            const pair< _Iterator1, _Iterator2 > & __x,
            const pair< _U1, _U2 > & __y)  [related]
```

Two pairs are equal iff their members are equal.

swap() [1/2]

```
enable_if< __and_< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type
swap (
            pair< _Iterator1, _Iterator2 > & __x,
            pair< _Iterator1, _Iterator2 > & __y)  [related]
```

Swap overload for pairs. Calls std::pair::swap().

Note

This std::swap overload is not declared in C++03 mode, which has performance implications, e.g. see https://gcc.gnu.org/PR38466

swap() [2/2]

```
enable_if< __and_< __is_swappable< _Iterator1 >, __is_swappable< _Iterator2 > >::value >::type
swap (
            pair< _Iterator1, _Iterator2 > & __x,
            pair< _Iterator1, _Iterator2 > & __y)  [related]
```

Swap overload for pairs. Calls std::pair::swap().

Note

This std::swap overload is not declared in C++03 mode, which has performance implications, e.g. see https://gcc.gnu.org/PR38466

6.96.5 Member Data Documentation

first [1/2]

```
_Iterator1 std::pair< _Iterator1, _Iterator2 >::first  [inherited]
```

The first member.

first [2/2]

```
_Iterator1 std::pair< _Iterator1, _Iterator2 >::first  [inherited]
```

The first member.

second [1/2]

```
_Iterator2 std::pair< _Iterator1, _Iterator2 >::second  [inherited]
```

The second member.

second [2/2]

```
_Iterator2 std::pair< _Iterator1, _Iterator2 >::second  [inherited]
```

The second member.

The documentation for this class was generated from the following file:

- iterator.h

6.97 __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory > Class Template Reference

`#include <iterator.h>`

Public Types

- typedef std::iterator_traits< _Iterator1 >::difference_type **difference_type**
- typedef _IteratorCategory **iterator_category**
- typedef _IteratorTriple * **pointer**
- typedef _IteratorTriple & **reference**
- typedef void **value_type**

Public Member Functions

- **_IteratorTriple** (const _Iterator1 &__first, const _Iterator2 &__second, const _Iterator3 &__third)
- **operator _Iterator3** () const
- _IteratorTriple **operator+** (difference_type __delta) const
- _IteratorTriple & **operator++** ()
- const _IteratorTriple **operator++** (int)
- difference_type **operator-** (const _IteratorTriple &__other) const
- _IteratorTriple & **operator--** ()
- const _IteratorTriple **operator--** (int)
- _IteratorTriple & **operator=** (const _IteratorTriple &__other)

Public Attributes

- _Iterator1 **_M_first**
- _Iterator2 **_M_second**
- _Iterator3 **_M_third**

6.97.1 Detailed Description

template<typename _Iterator1, typename _Iterator2, typename _Iterator3, typename _IteratorCategory>
class __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >

A triple of iterators. The usual iterator operations are applied to all three child iterators.
The documentation for this class was generated from the following file:

- iterator.h

6.98 __gnu_parallel::_Job< _DifferenceTp > Struct Template Reference

`#include <workstealing.h>`

Public Types

- typedef _DifferenceTp **_DifferenceType**

Public Attributes

- volatile _DifferenceType _M_first
- volatile _DifferenceType _M_last
- volatile _DifferenceType _M_load

6.98.1 Detailed Description

template<typename _DifferenceTp>
struct __gnu_parallel::_Job< _DifferenceTp >

One __job for a certain thread.

6.98.2 Member Data Documentation

_M_first

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_first
```

First element.
Changed by owning and stealing thread. By stealing thread, always incremented.
Referenced by __gnu_parallel::__for_each_template_random_access_workstealing().

_M_last

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_last
```

Last element.
Changed by owning thread only.
Referenced by __gnu_parallel::__for_each_template_random_access_workstealing().

_M_load

```
template<typename _DifferenceTp>
volatile _DifferenceType __gnu_parallel::_Job< _DifferenceTp >::_M_load
```

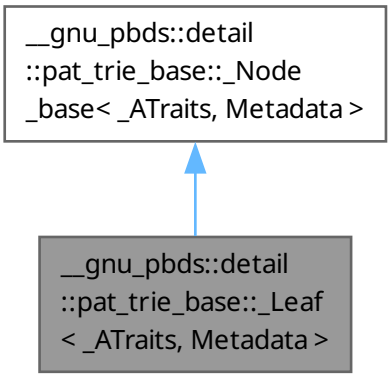
Number of elements, i.e. _M_last-_M_first+1.
Changed by owning thread only.
Referenced by __gnu_parallel::__for_each_template_random_access_workstealing().
The documentation for this struct was generated from the following file:

- workstealing.h

6.99 __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata > Struct Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >:



Public Types

- typedef _ATraits::const_iterator **a_const_iterator**
- typedef detail::rebind_traits< _Alloc, _ATraits >::const_pointer **a_const_pointer**
- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef _Node_base< _ATraits, Metadata > **base_type**
- typedef type_traits::const_reference **const_reference**
- typedef detail::rebind_traits< _Alloc, _Node_base >::pointer **node_pointer**
- typedef type_traits::reference **reference**
- typedef base_type::type_traits **type_traits**
- typedef type_traits::value_type **value_type**

Public Member Functions

- **_Leaf** (const_reference other)
- reference **value** ()
- const_reference **value** () const

Public Attributes

- node_pointer **m_p_parent**
- const node_type **m_type**

6.99.1 Detailed Description

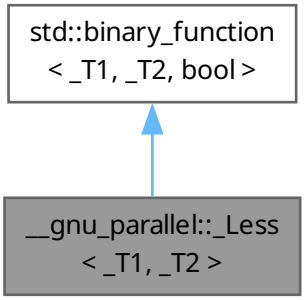
template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >

Leaf node for PATRICIA tree.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

6.100 __gnu_parallel::_Less< _T1, _T2 > Struct Template Reference

`#include <base.h>`
Inheritance diagram for __gnu_parallel::_Less< _T1, _T2 >:



Public Types

- typedef _T1 first_argument_type
- typedef bool result_type
- typedef _T2 second_argument_type

Public Member Functions

- bool **operator()** (const _T1 &__t1, const _T2 &__t2) const
- bool **operator()** (const _T2 &__t2, const _T1 &__t1) const

6.100.1 Detailed Description

template<typename _T1, typename _T2>
struct __gnu_parallel::_Less< _T1, _T2 >

Similar to std::less, but allows two different types.

6.100.2 Member Typedef Documentation

first_argument_type

`typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _T1, _T2, bool >::result_type [inherited]`
`result_type` is the return type

second_argument_type

```
typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type  [inherited]
```

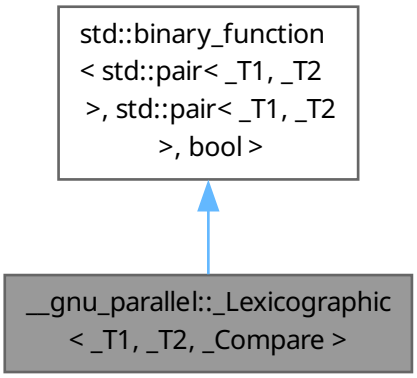
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- base.h

6.101 __gnu_parallel::_Lexicographic< _T1, _T2, _Compare > Class Template Reference

```
#include <multiseq_selection.h>
```

Inheritance diagram for __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >:



Public Types

- typedef std::pair< _T1, _T2 > first_argument_type
- typedef bool result_type
- typedef std::pair< _T1, _T2 > second_argument_type

Public Member Functions

- **_Lexicographic** (_Compare &__comp)
- bool **operator()** (const std::pair< _T1, _T2 > &__p1, const std::pair< _T1, _T2 > &__p2) const

6.101.1 Detailed Description

template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >

Compare __a pair of types lexicographically, ascending.

6.101.2 Member Typedef Documentation

first_argument_type

```
typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2
>, bool >::first_argument_type  [inherited]
```

first_argument_type is the type of the first argument

result_type

typedef bool std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::↩
result_type [inherited]
result_type is the return type

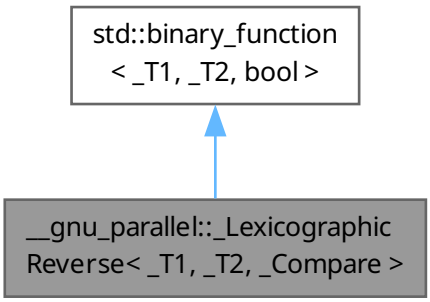
second_argument_type

typedef std::pair< _T1, _T2 > std::binary_function< std::pair< _T1, _T2 >, std::pair< _T1, _T2 >, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this class was generated from the following file:

- multiseq_selection.h

6.102 __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare > Class Template Reference

#include <multiseq_selection.h>
Inheritance diagram for __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >:



Public Types

- typedef _T1 first_argument_type
- typedef bool result_type
- typedef _T2 second_argument_type

Public Member Functions

- **_LexicographicReverse** (_Compare &__comp)
- bool **operator()** (const std::pair< _T1, _T2 > &__p1, const std::pair< _T1, _T2 > &__p2) const

6.102.1 Detailed Description

template<typename _T1, typename _T2, typename _Compare>
class __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >

Compare __a pair of types lexicographically, descending.

6.102.2 Member Typedef Documentation

first_argument_type

`typedef _T1 std::binary_function< _T1, _T2, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _T1, _T2, bool >::result_type [inherited]`
`result_type` is the return type

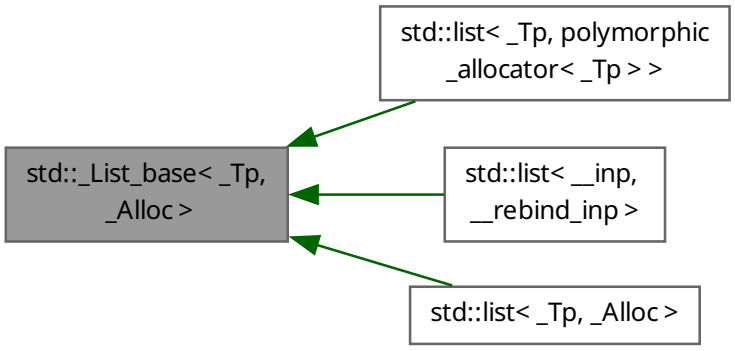
second_argument_type

`typedef _T2 std::binary_function< _T1, _T2, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this class was generated from the following file:

- multiseq_selection.h

6.103 std::_List_base< _Tp, _Alloc > Class Template Reference

`#include <stl_list.h>`
Inheritance diagram for std::_List_base< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**

Public Member Functions

- **_List_base** (_List_base &&)=default
- **_List_base** (_List_base &&__x, _Node_alloc_type &&__a)
- **_List_base** (_Node_alloc_type &&__a)
- **_List_base** (_Node_alloc_type &&__a, _List_base &&__x)
- **_List_base** (const _Node_alloc_type &__a) noexcept
- void **_M_clear** () noexcept
- size_t **_M_distance** (const __detail::_List_node_base *__first, const __detail::_List_node_base *__last) const
- const _Node_alloc_type & **_M_get_Node_allocator** () const noexcept
- _Node_alloc_type & **_M_get_Node_allocator** () noexcept
- void **_M_init** () noexcept
- void **_M_move_nodes** (_List_base &&__x)
- size_t **_M_node_count** () const

Static Public Member Functions

- static size_t **_S_distance** (const __detail::_List_node_base *__first, const __detail::_List_node_base *__last)

Protected Types

- typedef __gnu_cxx::__alloc_traits< _Node_alloc_type > **_Node_alloc_traits**
- typedef _Tp_alloc_traits::template rebind< typename_Node_traits::_Node >::other **_Node_alloc_type**
- using **_Node_ptr**
- typedef __list::_Node_traits< _Tp, typename _Tp_alloc_traits::pointer > **_Node_traits**
- typedef __gnu_cxx::__alloc_traits< _Tp_alloc_type > **_Tp_alloc_traits**
- typedef __gnu_cxx::__alloc_traits< _Alloc >::template rebind< _Tp >::other **_Tp_alloc_type**

Protected Member Functions

- void **_M_dec_size** (size_t __n)
- void **_M_destroy_node** (_Node_ptr __p)
- _Node_alloc_traits::pointer **_M_get_node** ()
- size_t **_M_get_size** () const
- void **_M_inc_size** (size_t __n)
- void **_M_put_node** (_Node_ptr __p) noexcept
- void **_M_set_size** (size_t __n)

Protected Attributes

- _List_impl **_M_impl**

6.103.1 Detailed Description

template<typename _Tp, typename _Alloc>
class std::_List_base< _Tp, _Alloc >

See bits/stl_deque.h's _Deque_base for an explanation.
The documentation for this class was generated from the following files:

- stl_list.h
- list.tcc

6.104 std::_List_const_iterator< _Tp > Struct Template Reference

`#include <stl_list.h>`

Public Types

- typedef const _List_node< _Tp > **_Node**
- typedef ptrdiff_t **difference_type**
- typedef _List_iterator< _Tp > **iterator**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef const _Tp * **pointer**
- typedef const _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- **_List_const_iterator** (const __detail::_List_node_base *__x) noexcept
- **_List_const_iterator** (const iterator &__x) noexcept
- iterator **_M_const_cast** () const noexcept
- reference **operator*** () const noexcept
- _List_const_iterator & **operator++** () noexcept
- _List_const_iterator **operator++** (int) noexcept
- _List_const_iterator & **operator--** () noexcept
- _List_const_iterator **operator--** (int) noexcept
- pointer **operator->** () const noexcept

Public Attributes

- const __detail::_List_node_base * **_M_node**

Friends

- bool **operator==** (const _List_const_iterator &__x, const _List_const_iterator &__y) noexcept

6.104.1 Detailed Description

template<typename _Tp>
struct std::_List_const_iterator< _Tp >

A list::const_iterator.
All the functions are op overloads.
The documentation for this struct was generated from the following files:

- stl_iterator_base_funcs.h
- stl_list.h

6.105 std::_List_iterator< _Tp > Struct Template Reference

`#include <stl_list.h>`

Public Types

- typedef _List_node< _Tp > **_Node**
- typedef ptrdiff_t **difference_type**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef _Tp * **pointer**
- typedef _Tp & **reference**
- typedef _Tp **value_type**

Public Member Functions

- **_List_iterator** (__detail::_List_node_base *__x) noexcept
- _List_iterator **_M_const_cast** () const noexcept
- reference **operator*** () const noexcept
- _List_iterator & **operator++** () noexcept
- _List_iterator **operator++** (int) noexcept
- _List_iterator & **operator--** () noexcept
- _List_iterator **operator--** (int) noexcept
- pointer **operator->** () const noexcept

Public Attributes

- __detail::_List_node_base * **_M_node**

Friends

- bool **operator==** (const _List_iterator &__x, const _List_iterator &__y) noexcept

6.105.1 Detailed Description

template<typename _Tp>
struct std::_List_iterator< _Tp >

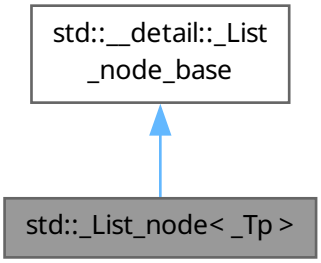
A list::iterator.
All the functions are op overloads.
The documentation for this struct was generated from the following files:

- stl_iterator_base_funcs.h
- stl_list.h

6.106 std::_List_node< _Tp > Struct Template Reference

`#include <stl_list.h>`

Inheritance diagram for std::_List_node< _Tp >:



Public Types

- typedef _List_node_base * **_Base_ptr**
- typedef _List_node * **_Node_ptr**

Public Member Functions

- _List_node_base * **_M_base** ()
- const _List_node_base * **_M_base** () const
- void **_M_hook** (_List_node_base *const __position) noexcept
- _Node_ptr **_M_node_ptr** ()
- void **_M_reverse** () noexcept
- void **_M_transfer** (_List_node_base *const __first, _List_node_base *const __last) noexcept
- void **_M_unhook** () noexcept
- _Tp * **_M_valptr** ()
- _Tp const * **_M_valptr** () const

Static Public Member Functions

- static void **swap** (_List_node_base &__x, _List_node_base &__y) noexcept

Public Attributes

- _List_node_base * **_M_next**
- _List_node_base * **_M_prev**
- __gnu_cxx::__aligned_membuf< _Tp > **_M_storage**

6.106.1 Detailed Description

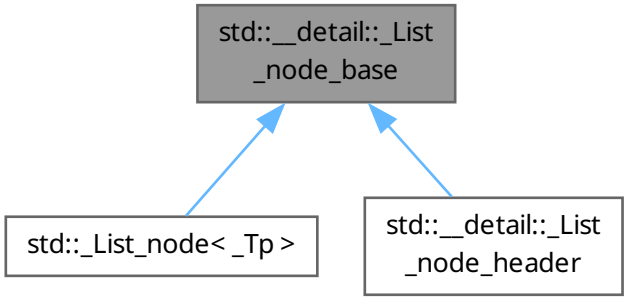
template<typename _Tp>
struct std::_List_node< _Tp >

An actual node in the list.
The documentation for this struct was generated from the following file:

- stl_list.h

6.107 std::__detail::_List_node_base Struct Reference

`#include <stl_list.h>`
Inheritance diagram for std::__detail::_List_node_base:



Public Types

- typedef _List_node_base * **_Base_ptr**

Public Member Functions

- _List_node_base * **_M_base** ()
- const _List_node_base * **_M_base** () const
- void **_M_hook** (_List_node_base *const __position) noexcept
- void **_M_reverse** () noexcept
- void **_M_transfer** (_List_node_base *const __first, _List_node_base *const __last) noexcept
- void **_M_unhook** () noexcept

Static Public Member Functions

- static void **swap** (_List_node_base &__x, _List_node_base &__y) noexcept

Public Attributes

- _List_node_base * **_M_next**
- _List_node_base * **_M_prev**

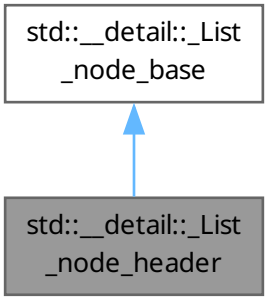
6.107.1 Detailed Description

Common part of a node in the list.
The documentation for this struct was generated from the following file:

- stl_list.h

6.108 std::__detail::_List_node_header Struct Reference

`#include <stl_list.h>`
Inheritance diagram for std::__detail::_List_node_header:



Public Types

- typedef _List_node_base * **_Base_ptr**

Public Member Functions

- **_List_node_header** (_List_node_header &&__x) noexcept
- _List_node_base * **_M_base** ()
- const _List_node_base * **_M_base** () const
- void **_M_hook** (_List_node_base *const __position) noexcept
- void **_M_init** () noexcept
- void **_M_move_nodes** (_List_node_header &&__x)
- void **_M_reverse** () noexcept
- void **_M_transfer** (_List_node_base *const __first, _List_node_base *const __last) noexcept
- void **_M_unhook** () noexcept

Static Public Member Functions

- static void **swap** (_List_node_base &__x, _List_node_base &__y) noexcept

Public Attributes

- _List_node_base * **_M_next**
- _List_node_base * **_M_prev**
- size_t **_M_size**

6.108.1 Detailed Description

The list node header.
The documentation for this struct was generated from the following file:

- stl_list.h

6.109 __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser Struct Reference

`#include <losertree.h>`

Public Attributes

- _Tp _M_key
- int _M_source
- bool _M_sup

6.109.1 Detailed Description

template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser

Internal representation of a _LoserTree element.

6.109.2 Member Data Documentation

_M_key

```
template<typename _Tp, typename _Compare>
_Tp __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_key
```

_M_key of the element in the _LoserTree.

_M_source

```
template<typename _Tp, typename _Compare>
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_source
```

__index of the __source __sequence.

_M_sup

```
template<typename _Tp, typename _Compare>
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser::_M_sup
```

flag, true iff this is a "maximum" __sentinel.
The documentation for this struct was generated from the following file:

- losertree.h

6.110 __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser Struct Reference

`#include <losertree.h>`

Public Attributes

- const _Tp * **_M_keyp**
- int **_M_source**
- bool **_M_sup**

6.110.1 Detailed Description

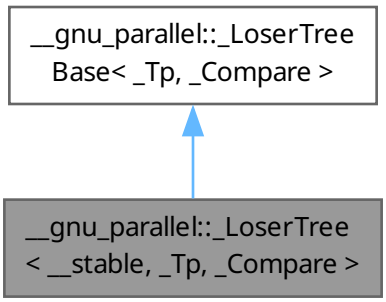
template<typename _Tp, typename _Compare>
struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser

Internal representation of _LoserTree __elements.
The documentation for this struct was generated from the following file:

- losertree.h

6.111 __gnu_parallel::_LoserTree< __stable, _Tp, _Compare > Class Template Reference

`#include <losertree.h>`
Inheritance diagram for __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >:



Public Member Functions

- **_LoserTree** (unsigned int __k, _Compare __comp)
- void __delete_min_insert (_Tp __key, bool __sup)
- int __get_min_source ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void __insert_start (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int _M_log_k
- unsigned int **_M_offset**

6.111.1 Detailed Description

template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >

Stable _LoserTree variant.
Provides the stable implementations of insert_start, __init_winner, __init and __delete_min_insert.
Unstable variant is done using partial specialisation below.

6.111.2 Member Function Documentation

__delete_min_insert()

```
template<bool __stable, typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >::__delete_min_insert (
```

```
            _Tp __key,
            bool __sup)  [inline]
```

Delete the smallest element and insert a new element from the previously smallest element's sequence.
This implementation is stable.

__get_min_source()

```
template<typename _Tp, typename _Compare>
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source ()  [inline], [inherited]
```

Returns

the index of the sequence with the smallest element.

References _M_losers.

__insert_start()

```
template<typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
            const _Tp & __key,
            int __source,
            bool __sup)  [inline], [inherited]
```

Initializes the sequence "_M_source" with the element "__key".

Parameters

__key	the element to insert
__source	__index of the __source __sequence
__sup	flag that determines whether the value to insert is an explicit __supremum.

References _M_first_insert, and _M_losers.

6.111.3 Member Data Documentation

_M_log_k

```
template<typename _Tp, typename _Compare>
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k  [protected], [inherited]
```

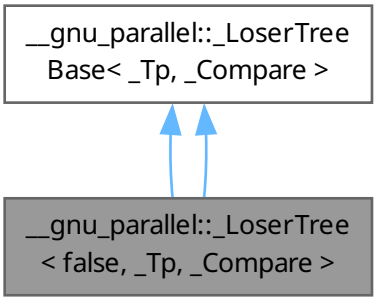
log_2{_M_k}
Referenced by _LoserTreeBase().
The documentation for this class was generated from the following file:

- losertree.h

6.112 __gnu_parallel::_LoserTree< false, _Tp, _Compare > Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for __gnu_parallel::_LoserTree< false, _Tp, _Compare >:



Public Member Functions

- **_LoserTree** (unsigned int __k, _Compare __comp)
- void __delete_min_insert (_Tp __key, bool __sup)
- int __get_min_source ()
- void **__init** ()
- unsigned int __init_winner (unsigned int __root)
- void __insert_start (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int _M_log_k
- unsigned int **_M_offset**

6.112.1 Detailed Description

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTree< false, _Tp, _Compare >

Unstable _LoserTree variant.
Stability (non-stable here) is selected with partial specialization.

6.112.2 Member Function Documentation

__delete_min_insert()

```
template<typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTree< false, _Tp, _Compare >::__delete_min_insert (
            _Tp __key,
            bool __sup)  [inline]
```

Delete the _M_key smallest element and insert the element __key instead.

Parameters

__key	the _M_key to insert
__sup	true iff __key is an explicitly marked supremum

__get_min_source()

```
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source () [inline]
```

Returns

the index of the sequence with the smallest element.

__init_winner()

```
template<typename _Tp, typename _Compare>
unsigned int __gnu_parallel::_LoserTree< false, _Tp, _Compare >::__init_winner (
            unsigned int __root) [inline]
```

Computes the winner of the competition at position "__root".
Called recursively (starting at 0) to build the initial tree.

Parameters

__root	__index of the "game" to start.

References __init_winner().
Referenced by __init_winner().

__insert_start()

```
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
            const _Tp & __key,
            int __source,
            bool __sup) [inline]
```

Initializes the sequence "_M_source" with the element "__key".

Parameters

__key	the element to insert
__source	__index of the __source __sequence
__sup	flag that determines whether the value to insert is an explicit __supremum.

6.112.3 Member Data Documentation

_M_log_k

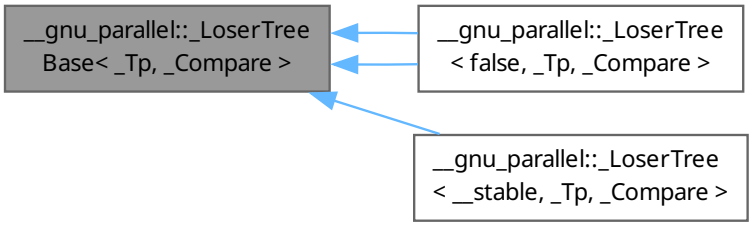
```
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k [protected]
```

log_2{_M_k}
The documentation for this class was generated from the following file:

- losertree.h

6.113 __gnu_parallel::_LoserTreeBase< _Tp, _Compare > Class Template Reference

`#include <losertree.h>`
Inheritance diagram for __gnu_parallel::_LoserTreeBase< _Tp, _Compare >:



Classes

- struct _Loser

Public Member Functions

- _LoserTreeBase (unsigned int __k, _Compare __comp)
- ~_LoserTreeBase ()
- int __get_min_source ()
- void __insert_start (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare _M_comp
- bool _M_first_insert
- unsigned int **_M_ik**
- unsigned int **_M_k**
- unsigned int _M_log_k
- _Loser * _M_losers
- unsigned int **_M_offset**

6.113.1 Detailed Description

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >

Guarded loser/tournament tree.
The smallest element is at the top.
Guarding is done explicitly through one flag _M_sup per element, inf is not needed due to a better initialization routine.
This is a well-performing variant.

Parameters

_Tp	the element type

_Compare	the comparator to use, defaults to std::less<_Tp>

6.113.2 Constructor & Destructor Documentation

_LoserTreeBase()

```
template<typename _Tp, typename _Compare>
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_LoserTreeBase (
            unsigned int __k,
            _Compare __comp)  [inline]
```

The constructor.

Parameters

__k	The number of sequences to merge.
__comp	The comparator to use.

References __gnu_parallel::__rd_log2(), _M_comp, _M_first_insert, _M_log_k, and _M_losers.

~_LoserTreeBase()

```
template<typename _Tp, typename _Compare>
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::~_LoserTreeBase ()  [inline]
```

The destructor.
References _M_losers.

6.113.3 Member Function Documentation

__get_min_source()

```
template<typename _Tp, typename _Compare>
int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__get_min_source ()  [inline]
```

Returns

the index of the sequence with the smallest element.

References _M_losers.

__insert_start()

```
template<typename _Tp, typename _Compare>
void __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::__insert_start (
            const _Tp & __key,
            int __source,
            bool __sup)  [inline]
```

Initializes the sequence "_M_source" with the element "__key".

Parameters

__key	the element to insert
__source	__index of the __source __sequence

__sup	flag that determines whether the value to insert is an explicit __supremum.

References _M_first_insert, and _M_losers.

6.113.4 Member Data Documentation

_M_comp

```
template<typename _Tp, typename _Compare>
_Compare __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_comp  [protected]
```

_Compare to use.
Referenced by _LoserTreeBase().

_M_first_insert

```
template<typename _Tp, typename _Compare>
bool __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_first_insert  [protected]
```

State flag that determines whether the _LoserTree is empty.
Only used for building the _LoserTree.
Referenced by _LoserTreeBase(), and __insert_start().

_M_log_k

```
template<typename _Tp, typename _Compare>
unsigned int __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_log_k  [protected]
```

log_2{_M_k}
Referenced by _LoserTreeBase().

_M_losers

```
template<typename _Tp, typename _Compare>
_Loser* __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_M_losers  [protected]
```

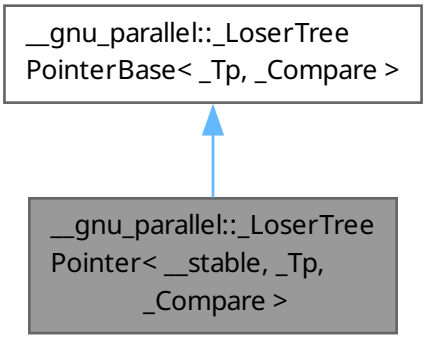
_LoserTree __elements.
Referenced by _LoserTreeBase(), ~_LoserTreeBase(), __get_min_source(), and __insert_start().
The documentation for this class was generated from the following file:

- losertree.h

6.114 __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare > Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointer** (unsigned int __k, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int **_M_offset**

6.114.1 Detailed Description

template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >

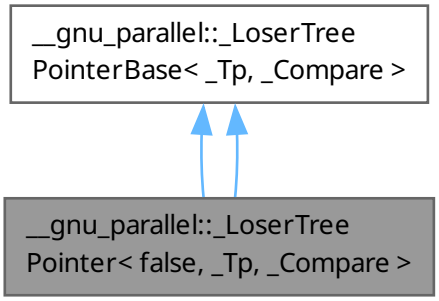
Stable _LoserTree implementation.
The unstable variant is implemented using partial instantiation below.
The documentation for this class was generated from the following file:

- losertree.h

6.115 __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare > Class Template Reference

`#include <losertree.h>`

Inheritance diagram for __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointer** (unsigned int __k, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int **_M_offset**

6.115.1 Detailed Description

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >

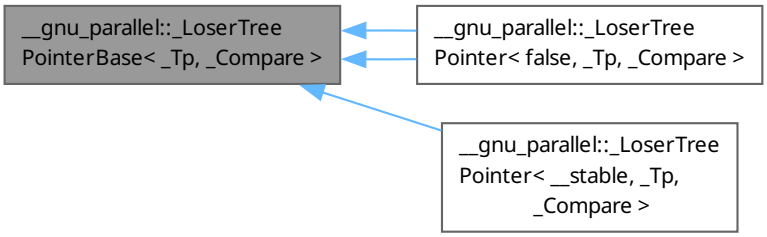
Unstable _LoserTree implementation.
The stable variant is above.
The documentation for this class was generated from the following file:

- losertree.h

6.116 __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare > Class Template Reference

`#include <losertree.h>`

Inheritance diagram for __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >:



Classes

- struct _Loser

Public Member Functions

- **_LoserTreePointerBase** (unsigned int __k, _Compare __comp=std::less< _Tp >())
- int **__get_min_source** ()
- void **__insert_start** (const _Tp &__key, int __source, bool __sup)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser * **_M_losers**
- unsigned int **_M_offset**

6.116.1 Detailed Description

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >

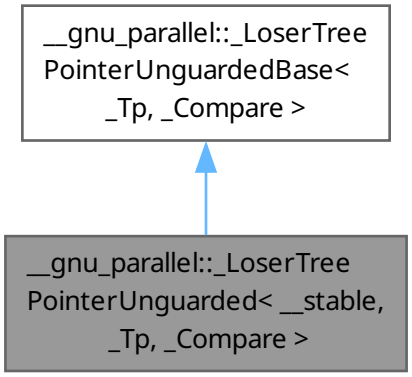
Base class of _Loser Tree implementation using pointers.
The documentation for this class was generated from the following file:

- losertree.h

6.117 __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare > Class Template Reference

`#include <losertree.h>`

Inheritance diagram for __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointerUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int **_M_offset**

6.117.1 Detailed Description

template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >

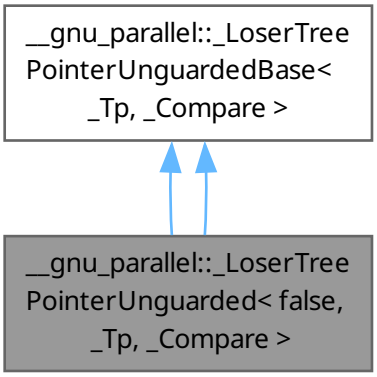
Stable unguarded _LoserTree variant storing pointers.
Unstable variant is implemented below using partial specialization.
The documentation for this class was generated from the following file:

- losertree.h

6.118 __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare > Class Template Reference

`#include <losertree.h>`

Inheritance diagram for __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointerUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (const _Tp &__key, bool __sup)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int **_M_offset**

6.118.1 Detailed Description

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >

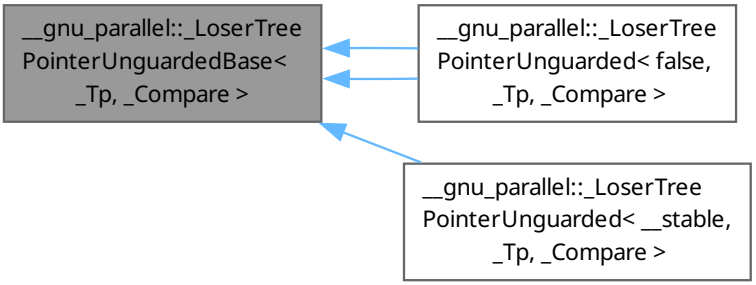
Unstable unguarded _LoserTree variant storing pointers.
Stable variant is above.
The documentation for this class was generated from the following file:

- losertree.h

6.119 __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare > Class Template Reference

`#include <losertree.h>`

Inheritance diagram for __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >:



Public Member Functions

- **_LoserTreePointerUnguardedBase** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- int **__get_min_source** ()
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser * **_M_losers**
- unsigned int **_M_offset**

6.119.1 Detailed Description

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >

Unguarded loser tree, keeping only pointers to the elements in the tree structure.
No guarding is done, therefore not a single input sequence must run empty. This is a very fast variant.
The documentation for this class was generated from the following file:

- losertree.h

6.120 __gnu_parallel::_LoserTreeTraits< _Tp > Struct Template Reference

`#include <multiway_merge.h>`

Static Public Attributes

- static const bool _M_use_pointer

6.120.1 Detailed Description

template<typename _Tp>
struct __gnu_parallel::_LoserTreeTraits< _Tp >

Traits for determining whether the loser tree should use pointers or copies.
The field "_M_use_pointer" is used to determine whether to use pointers in he loser trees or whether to copy the values into the loser tree.
The default behavior is to use pointers if the data type is 4 times as big as the pointer to it.
Specialize for your data type to customize the behavior.
Example:
template<> struct _LoserTreeTraits<int> { static const bool _M_use_pointer = false; };
template<> struct _LoserTreeTraits<heavyweight_type> { static const bool _M_use_pointer = true; };

Parameters

_Tp	type to give the loser tree traits for.

6.120.2 Member Data Documentation

_M_use_pointer

```
template<typename _Tp>
const bool __gnu_parallel::_LoserTreeTraits< _Tp >::_M_use_pointer  [static]
```

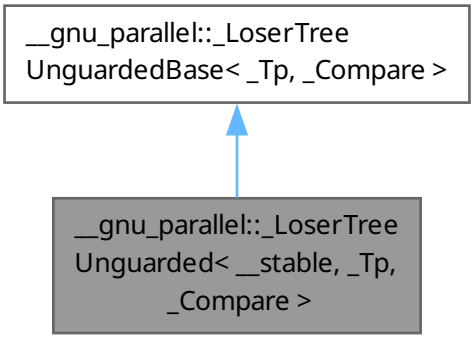
True iff to use pointers instead of values in loser trees.
The default behavior is to use pointers if the data type is four times as big as the pointer to it.
The documentation for this struct was generated from the following file:

- multiway_merge.h

6.121 __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare > Class Template Reference

```
#include <losertree.h>
```

Inheritance diagram for __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >:



Public Member Functions

- **_LoserTreeUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (_Tp __key, bool)
- int **__get_min_source** ()
- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int **_M_offset**

6.121.1 Detailed Description

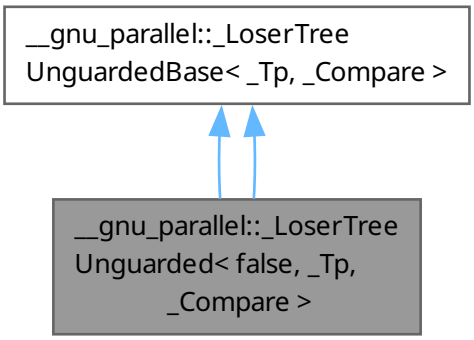
template<bool __stable, typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >

Stable implementation of unguarded _LoserTree.
Unstable variant is selected below with partial specialization.
The documentation for this class was generated from the following file:

- losertree.h

6.122 __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare > Class Template Reference

`#include <losertree.h>`
Inheritance diagram for __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >:



Public Member Functions

- **_LoserTreeUnguarded** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- void **__delete_min_insert** (_Tp __key, bool)
- int **__get_min_source** ()

- void **__init** ()
- unsigned int **__init_winner** (unsigned int __root)
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- unsigned int **_M_ik**
- unsigned int **_M_offset**

6.122.1 Detailed Description

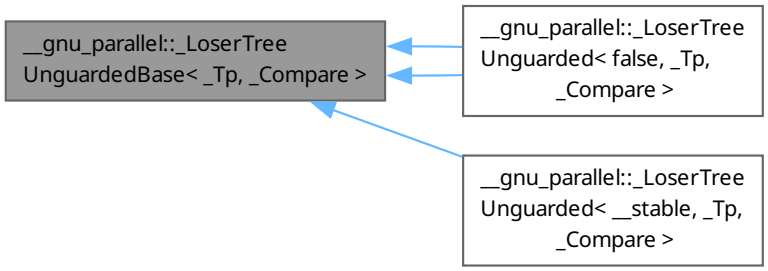
template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >

Non-Stable implementation of unguarded _LoserTree.
Stable implementation is above.
The documentation for this class was generated from the following file:

- losertree.h

6.123 __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare > Class Template Reference

`#include <losertree.h>`
Inheritance diagram for __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >:



Public Member Functions

- **_LoserTreeUnguardedBase** (unsigned int __k, const _Tp &__sentinel, _Compare __comp=std::less< _Tp >())
- int **__get_min_source** ()
- void **__insert_start** (const _Tp &__key, int __source, bool)

Protected Attributes

- _Compare **_M_comp**
- unsigned int **_M_ik**
- unsigned int **_M_k**
- _Loser * **_M_losers**
- unsigned int **_M_offset**

6.123.1 Detailed Description

template<typename _Tp, typename _Compare>
class __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >

Base class for unguarded _LoserTree implementation.
The whole element is copied into the tree structure.
No guarding is done, therefore not a single input sequence must run empty. Unused __sequence heads are marked with a sentinel which is > all elements that are to be merged.
This is a very fast variant.
The documentation for this class was generated from the following file:

- losertree.h

6.124 __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc > Struct Template Reference

`#include <pat_trie_base.hpp>`

Public Types

- typedef _Alloc **allocator_type**
- typedef detail::rebind_traits< _Alloc, Metadata >::const_reference **const_reference**
- typedef Metadata **metadata_type**

Public Member Functions

- const_reference **get_metadata** () const

Public Attributes

- metadata_type **m_metadata**

6.124.1 Detailed Description

template<typename Metadata, typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >

Metadata base primary template.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

6.125 __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc > Struct Template Reference

`#include <pat_trie_base.hpp>`

Public Types

- typedef _Alloc **allocator_type**
- typedef detail::rebind_traits< _Alloc, null_type >::const_reference **const_reference**
- typedef null_type **metadata_type**

Public Member Functions

- const_reference **get_metadata** () const

Public Attributes

- metadata_type **m_metadata**

6.125.1 Detailed Description

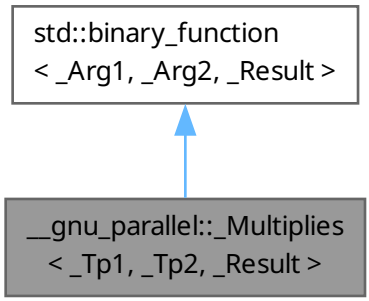
template<typename _Alloc>
struct __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >

Specialization for null metadata.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

6.126 __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result > Struct Template Reference

`#include <base.h>`
Inheritance diagram for __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >:



Public Types

- typedef _Arg1 first_argument_type
- typedef _Result result_type
- typedef _Arg2 second_argument_type

Public Member Functions

- _Result **operator()** (const _Tp1 &__x, const _Tp2 &__y) const

6.126.1 Detailed Description

template<typename _Tp1, typename _Tp2, typename _Result = __typeof__(*static_cast<_Tp1*>(0) * *static↩ _cast<_Tp2*>(0))>
struct __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >

Similar to std::multiplies, but allows two different types.

6.126.2 Member Typedef Documentation

first_argument_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

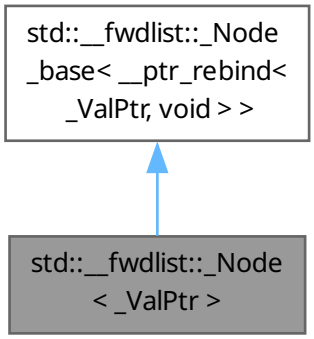
```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- base.h

6.127 std::__fwdlist::_Node< _ValPtr > Struct Template Reference

`#include <forward_list.h>`
Inheritance diagram for std::__fwdlist::_Node< _ValPtr >:



Public Types

- using **_Base_ptr**
- using **_Node_ptr**
- using **value_type**

Public Member Functions

- **_Node** (_Node &&)=delete

- _Base_ptr **_M_base_ptr** () const
- _Node_ptr **_M_node_ptr** ()
- void **_M_reverse_after** () noexcept
- _Base_ptr **_M_transfer_after** (_Base_ptr __begin, _Base_ptr __end) noexcept
- const value_type * **_M_valptr** () const noexcept
- value_type * **_M_valptr** () noexcept

Public Attributes

- _Base_ptr **_M_next**
- _Uninit_storage **_M_u**

6.127.1 Detailed Description

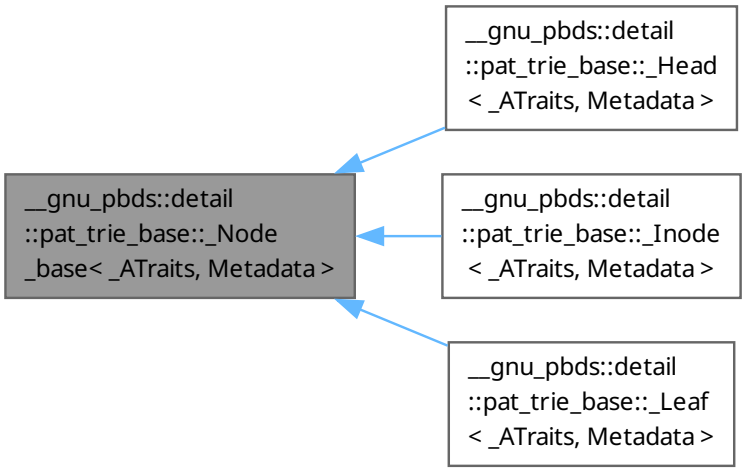
template<typename _ValPtr>
struct std::__fwdlist::_Node< _ValPtr >

A helper node class for forward_list.
The documentation for this struct was generated from the following file:

- forward_list.h

6.128 __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata > Struct Template Reference

`#include <pat_trie_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >:



Public Types

- typedef _ATraits::const_iterator **a_const_iterator**
- typedef detail::rebind_traits< _Alloc, _ATraits >::const_pointer **a_const_pointer**

- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef detail::rebind_traits< _Alloc, _Node_base >::pointer **node_pointer**
- typedef _ATraits::type_traits **type_traits**

Public Member Functions

- **_Node_base** (node_type type)

Public Attributes

- node_pointer **m_p_parent**
- const node_type **m_type**

6.128.1 Detailed Description

template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >

Node base.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

6.129 std::__fwdlist::_Node_base< _VoidPtr > Struct Template Reference

`#include <forward_list.h>`

Public Types

- using **_Base_ptr**

Public Member Functions

- **_Node_base** (_Node_base &&__x) noexcept
- **_Node_base** (const _Node_base &)=delete
- _Base_ptr **_M_base_ptr** () const
- void **_M_reverse_after** () noexcept
- _Base_ptr **_M_transfer_after** (_Base_ptr __begin, _Base_ptr __end) noexcept
- _Node_base & **operator=** (_Node_base &&__x) noexcept
- _Node_base & **operator=** (const _Node_base &)=delete

Public Attributes

- _Base_ptr **_M_next**

6.129.1 Detailed Description

template<typename _VoidPtr>
struct std::__fwdlist::_Node_base< _VoidPtr >

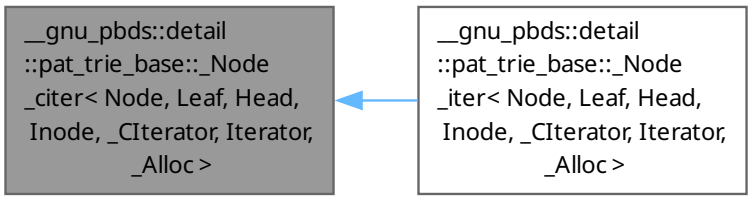
The node-base type for allocators that use fancy pointers.
The documentation for this struct was generated from the following file:

- forward_list.h

6.130 __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference

`#include <pat_trie_base.hpp>`

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >:



Public Types

- typedef value_type **const_reference**
- typedef trivial_iterator_difference_type **difference_type**
- typedef trivial_iterator_tag **iterator_category**
- typedef rebind_traits< _Alloc, metadata_type >::const_reference metadata_const_reference
- typedef Node::metadata_type metadata_type
- typedef value_type **reference**
- typedef _Alloc::size_type **size_type**
- typedef _CIterator **value_type**

Public Member Functions

- **_Node_citer** (node_pointer p_nd=0, a_const_pointer p_traits=0)
- _Node_citer get_child (size_type i) const
- metadata_const_reference get_metadata () const
- size_type num_children () const
- bool operator!= (const _Node_citer &other) const
- const_reference operator* () const
- bool operator== (const _Node_citer &other) const
- std::pair< a_const_iterator, a_const_iterator > valid_prefix () const

Public Attributes

- node_pointer **m_p_nd**
- a_const_pointer **m_p_traits**

Protected Types

- typedef Node::a_const_iterator **a_const_iterator**
- typedef Node::a_const_pointer **a_const_pointer**
- typedef rebind_traits< _Alloc, Inode >::const_pointer **inode_const_pointer**
- typedef rebind_traits< _Alloc, Inode >::pointer **inode_pointer**

- typedef rebind_traits< _Alloc, Leaf >::const_pointer **leaf_const_pointer**
- typedef rebind_traits< _Alloc, Leaf >::pointer **leaf_pointer**
- typedef rebind_traits< _Alloc, Node >::pointer **node_pointer**

6.130.1 Detailed Description

template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>
class __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >

Node const iterator.

6.130.2 Member Typedef Documentation

metadata_const_reference

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
typedef rebind_traits<_Alloc,metadata_type>::const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::metadata_const_reference
```

Const metadata reference type.

metadata_type

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head,
Inode, _CIterator, Iterator, _Alloc >::metadata_type
```

Metadata type.

6.130.3 Member Function Documentation

get_child()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
_Node_citer __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::get_child (
            size_type i) const  [inline]
```

Returns a __const node __iterator to the corresponding node's i-th child.

get_metadata()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::get_metadata () const  [inline]
```

Metadata access.

num_children()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::num_children () const  [inline]
```

Returns the number of children in the corresponding node.
Referenced by __gnu_pbds::detail::pat_trie_base::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >::operator*().

operator"!=()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator!= (
            const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const  [inline]
```

Compares content (negatively) to a different iterator object.

operator*()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _↩
CIterator, Iterator, _Alloc >::operator* () const  [inline]
```

Const access; returns the __const iterator* associated with the current leaf.

operator==()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator== (
            const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const  [inline]
```

Compares content to a different iterator object.

valid_prefix()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
std::pair< a_const_iterator, a_const_iterator > __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const  [inline]
```

Subtree valid prefix.

The documentation for this class was generated from the following file:

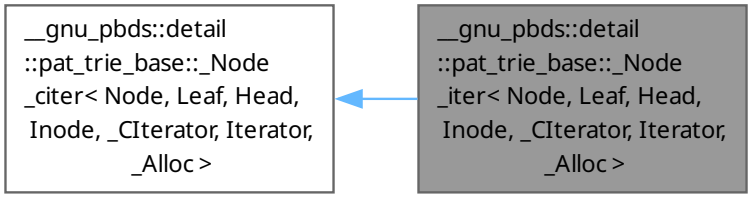
- pat_trie_base.hpp

6.131 __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > Class Template Reference

```
#include <pat_trie_base.hpp>
```

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator,

_Alloc >:



Public Types

- typedef value_type **const_reference**
- typedef trivial_iterator_difference_type **difference_type**
- typedef trivial_iterator_tag **iterator_category**
- typedef rebind_traits< _Alloc, metadata_type >::const_reference metadata_const_reference
- typedef Node::metadata_type metadata_type
- typedef value_type **reference**
- typedef base_type::size_type **size_type**
- typedef Iterator **value_type**

Public Member Functions

- **_Node_iter** (node_pointer p_nd=0, a_const_pointer p_traits=0)
- _Node_iter get_child (size_type i) const
- metadata_const_reference get_metadata () const
- size_type num_children () const
- bool operator!= (const _Node_citer &other) const
- reference operator* () const
- bool operator== (const _Node_citer &other) const
- std::pair< a_const_iterator, a_const_iterator > valid_prefix () const

Public Attributes

- node_pointer **m_p_nd**
- a_const_pointer **m_p_traits**

Protected Types

- typedef Node::a_const_iterator **a_const_iterator**
- typedef rebind_traits< _Alloc, Inode >::const_pointer **inode_const_pointer**
- typedef rebind_traits< _Alloc, Leaf >::const_pointer **leaf_const_pointer**
- typedef rebind_traits< _Alloc, Leaf >::pointer **leaf_pointer**

6.131.1 Detailed Description

template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename Iterator, typename _Alloc>
class __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >

Node iterator.

6.131.2 Member Typedef Documentation

metadata_const_reference

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
typedef rebind_traits<_Alloc,metadata_type>::const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::metadata_const_reference  [inherited]
```

Const metadata reference type.

metadata_type

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
typedef Node::metadata_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head,
Inode, _CIterator, Iterator, _Alloc >::metadata_type  [inherited]
```

Metadata type.

6.131.3 Member Function Documentation

get_child()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
_Node_iter __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::get_child (
            size_type i) const  [inline]
```

Returns a node __iterator to the corresponding node's i-th child.

get_metadata()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
metadata_const_reference __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode,
_CIterator, Iterator, _Alloc >::get_metadata () const  [inline], [inherited]
```

Metadata access.

num_children()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
size_type __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::num_children () const  [inline], [inherited]
```

Returns the number of children in the corresponding node.
Referenced by __gnu_pbds::detail::pat_trie_base::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc >::operator*().

operator"!=()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator!= (
            const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const  [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

operator*()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
reference __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator,
Iterator, _Alloc >::operator* () const  [inline]
```

Access; returns the iterator* associated with the current leaf.

operator==()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
bool __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator,
_Alloc >::operator== (
            const _Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc > & other)
const  [inline], [inherited]
```

Compares content to a different iterator object.

valid_prefix()

```
template<typename Node, typename Leaf, typename Head, typename Inode, typename _CIterator, typename
Iterator, typename _Alloc>
std::pair< a_const_iterator, a_const_iterator > __gnu_pbds::detail::pat_trie_base::_Node_citer<
Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >::valid_prefix () const  [inline], [inherited]
```

Subtree valid prefix.

The documentation for this class was generated from the following file:

- pat_trie_base.hpp

6.132 __gnu_debug::_Not_equal_to< _Type > Class Template Reference

```
#include <safe_sequence.h>
```

Public Member Functions

- constexpr **_Not_equal_to** (const _Type &__v)
- bool **operator()** (const _Type &__x) const

6.132.1 Detailed Description

template<typename _Type>
class __gnu_debug::_Not_equal_to< _Type >

A simple function object that returns true if the passed-in value is not equal to the stored value. It saves typing over using both bind1st and not_equal.

The documentation for this class was generated from the following file:

- safe_sequence.h

6.133 std::_Not_fn< _Fn > Class Template Reference

`#include <functional>`

Public Member Functions

- template<typename _Fn2>
  constexpr **_Not_fn** (_Fn2 &&__fn, int)
- **_Not_fn** (_Not_fn &&__fn)=default
- **_Not_fn** (const _Not_fn &__fn)=default
- template<typename... _Args, typename = enable_if_t<!__is_invocable<_Fn &&, _Args...>::value>>
  void **operator()** (_Args &&... __args) &&=delete
- template<typename... _Args, typename = enable_if_t<__is_invocable<_Fn &&, _Args...>::value>>
  constexpr decltype(_S_not< __inv_res_t< _Fn &&, _Args... > >()) **operator()** (_Args &&... __args) &&noexcept(__is_nothrow_invocable< _Fn &&, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn &&, _Args... > >()))
- template<typename... _Args, typename = enable_if_t<!__is_invocable<_Fn &, _Args...>::value>>
  void **operator()** (_Args &&... __args) &=delete
- template<typename... _Args, typename = enable_if_t<__is_invocable<_Fn &, _Args...>::value>>
  constexpr decltype(_S_not< __inv_res_t< _Fn &, _Args... > >()) **operator()** (_Args &&... __args) &noexcept(__is_nothrow_invocable< _Fn &, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn &, _Args... > >()))
- template<typename... _Args, typename = enable_if_t<!__is_invocable<_Fn const &&, _Args...>::value>>
  void **operator()** (_Args &&... __args) const &&=delete
- template<typename... _Args, typename = enable_if_t<__is_invocable<_Fn const &&, _Args...>::value>>
  constexpr decltype(_S_not< __inv_res_t< _Fn const &&, _Args... > >()) **operator()** (_Args &&... __args) const &&noexcept(__is_nothrow_invocable< _Fn const &&, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn const &&, _Args... > >()))
- template<typename... _Args, typename = enable_if_t<!__is_invocable<_Fn const &, _Args...>::value>>
  void **operator()** (_Args &&... __args) const &=delete
- template<typename... _Args, typename = enable_if_t<__is_invocable<_Fn const &, _Args...>::value>>
  constexpr decltype(_S_not< __inv_res_t< _Fn const &, _Args... > >()) **operator()** (_Args &&... __args) const &noexcept(__is_nothrow_invocable< _Fn const &, _Args... >::value &&noexcept(_S_not< __inv_res_t< _Fn const &, _Args... > >()))

6.133.1 Detailed Description

template<typename _Fn>
class std::_Not_fn< _Fn >

Generalized negator.
The documentation for this class was generated from the following file:

- functional

6.134 __gnu_parallel::_Nothing Struct Reference

`#include <for_each_selectors.h>`

Public Member Functions

- template<typename _It>
  void operator() (_It __i)

6.134.1 Detailed Description

Functor doing nothing.
For some __reduction tasks (this is not a function object, but is passed as __selector __dummy parameter.

6.134.2 Member Function Documentation

operator()()

```
template<typename _It>
void __gnu_parallel::_Nothing::operator() (
             _It __i)  [inline]
```

Functor execution.

Parameters

↩ _↩ ↩ _↩ *i*	iterator referencing object.

The documentation for this struct was generated from the following file:

- for_each_selectors.h

6.135 __gnu_parallel::_Piece< _DifferenceTp > Struct Template Reference

```
#include <multiway_mergesort.h>
```

Public Types

- typedef _DifferenceTp **_DifferenceType**

Public Attributes

- _DifferenceType _M_begin
- _DifferenceType _M_end

6.135.1 Detailed Description

template<typename _DifferenceTp>
struct __gnu_parallel::_Piece< _DifferenceTp >

Subsequence description.

6.135.2 Member Data Documentation

_M_begin

```
template<typename _DifferenceTp>
_DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_begin
```

Begin of subsequence.

_M_end

```
template<typename _DifferenceTp>
_DifferenceType __gnu_parallel::_Piece< _DifferenceTp >::_M_end
```

End of subsequence.

The documentation for this struct was generated from the following file:

- multiway_mergesort.h

6.136 std::_Placeholder< _Num > Struct Template Reference

```
#include <functional>
```

6.136.1 Detailed Description

template<int _Num>
struct std::_Placeholder< _Num >

The type of placeholder objects defined by libstdc++.

Since

C++11

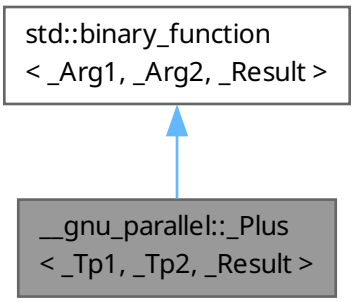
The documentation for this struct was generated from the following file:

- functional

6.137 __gnu_parallel::_Plus< _Tp1, _Tp2, _Result > Struct Template Reference

```
#include <base.h>
```

Inheritance diagram for __gnu_parallel::_Plus< _Tp1, _Tp2, _Result >:



Public Types

- typedef _Arg1 first_argument_type
- typedef _Result result_type
- typedef _Arg2 second_argument_type

Public Member Functions

- _Result **operator()** (const _Tp1 &__x, const _Tp2 &__y) const

6.137.1 Detailed Description

template<typename _Tp1, typename _Tp2, typename _Result = __typeof__(*static_cast<_Tp1*>(0) + *static_cast<_Tp2*>(0))>
struct __gnu_parallel::_Plus< _Tp1, _Tp2, _Result >

Similar to std::plus, but allows two different types.

6.137.2 Member Typedef Documentation

first_argument_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- base.h

6.138 __gnu_parallel::_PMWMSSortingData< _RAIter > Struct Template Reference

`#include <multiway_mergesort.h>`

Public Types

- typedef _TraitsType::difference_type **_DifferenceType**
- typedef std::iterator_traits< _RAIter > **_TraitsType**
- typedef _TraitsType::value_type **_ValueType**

Public Attributes

- _ThreadIndex _M_num_threads
- _DifferenceType * _M_offsets
- std::vector< _Piece< _DifferenceType > > * _M_pieces
- _ValueType * _M_samples
- _RAIter _M_source
- _DifferenceType * _M_starts
- _ValueType ** _M_temporary

6.138.1 Detailed Description

template<typename _RAIter>
struct __gnu_parallel::_PMWMSSortingData< _RAIter >

Data accessed by all threads.
PMWMS = parallel multiway mergesort

6.138.2 Member Data Documentation

_M_num_threads

```
template<typename _RAIter>
_ThreadIndex __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_num_threads
```

Number of threads involved.
Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

_M_offsets

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_offsets
```

Offsets to add to the found positions.
Referenced by __gnu_parallel::parallel_sort_mwms().

_M_pieces

```
template<typename _RAIter>
std::vector<_Piece<_DifferenceType> >* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_pieces
```

Pieces of data to merge [thread][__sequence].
Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

_M_samples

```
template<typename _RAIter>
_ValueType* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_samples
```

Samples.
Referenced by __gnu_parallel::__determine_samples(), and __gnu_parallel::parallel_sort_mwms().

_M_source

```
template<typename _RAIter>
_RAIter __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_source
```

Input __begin.
Referenced by __gnu_parallel::__determine_samples(), __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_

_M_starts

```
template<typename _RAIter>
_DifferenceType* __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_starts
```

Start indices, per thread.
Referenced by __gnu_parallel::__determine_samples(), __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_

_M_temporary

```
template<typename _RAIter>
_ValueType** __gnu_parallel::_PMWMSSortingData< _RAIter >::_M_temporary
```

Storage in which to sort.
Referenced by __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().
The documentation for this struct was generated from the following file:

- multiway_mergesort.h

6.139 __gnu_cxx::_Pointer_adapter< _Storage_policy > Class Template Reference

`#include <pointer.h>`

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef _Storage_policy::element_type **element_type**
- typedef std::random_access_iterator_tag **iterator_category**
- typedef _Pointer_adapter **pointer**
- typedef _Reference_type< element_type >::reference **reference**
- typedef _Unqualified_type< element_type >::type **value_type**

Public Member Functions

- template<typename _Up>
  _Pointer_adapter (_Up *__arg)
- **_Pointer_adapter** (const _Pointer_adapter &__arg)
- template<typename _Up>
  _Pointer_adapter (const _Pointer_adapter< _Up > &__arg)
- **_Pointer_adapter** (element_type *__arg=0)
- **operator bool** () const
- reference **operator*** () const
- _Pointer_adapter & **operator++** ()
- _Pointer_adapter **operator++** (int)
- _Pointer_adapter & **operator+=** (int __offset)
- _Pointer_adapter & **operator+=** (long __offset)
- _Pointer_adapter & **operator+=** (long long __offset)
- _Pointer_adapter & **operator+=** (short __offset)
- _Pointer_adapter & **operator+=** (unsigned int __offset)
- _Pointer_adapter & **operator+=** (unsigned long __offset)
- _Pointer_adapter & **operator+=** (unsigned long long __offset)
- _Pointer_adapter & **operator+=** (unsigned short __offset)
- template<typename _Up>
  std::ptrdiff_t **operator-** (const _Pointer_adapter< _Up > &__rhs) const
- _Pointer_adapter & **operator--** ()
- _Pointer_adapter **operator--** (int)
- _Pointer_adapter & **operator-=** (int __offset)
- _Pointer_adapter & **operator-=** (long __offset)
- _Pointer_adapter & **operator-=** (long long __offset)
- _Pointer_adapter & **operator-=** (short __offset)
- _Pointer_adapter & **operator-=** (unsigned int __offset)
- _Pointer_adapter & **operator-=** (unsigned long __offset)
- _Pointer_adapter & **operator-=** (unsigned long long __offset)
- _Pointer_adapter & **operator-=** (unsigned short __offset)
- element_type * **operator->** () const

- template<typename _Up>
  _Pointer_adapter & **operator=** (_Up *__arg)
- _Pointer_adapter & **operator=** (const _Pointer_adapter &__arg)
- template<typename _Up>
  _Pointer_adapter & **operator=** (const _Pointer_adapter< _Up > &__arg)
- reference **operator[ ]** (std::ptrdiff_t __index) const

Friends

- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, int __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, long __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, long long __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, short __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, unsigned int __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, unsigned long __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, unsigned long long __offset)
- _Pointer_adapter **operator+** (const _Pointer_adapter &__lhs, unsigned short __offset)
- _Pointer_adapter **operator+** (int __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (long __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (long long __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (short __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (unsigned int __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (unsigned long __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (unsigned long long __offset, const _Pointer_adapter &__rhs)
- _Pointer_adapter **operator+** (unsigned short __offset, const _Pointer_adapter &__rhs)
- template<typename _Up>
  std::ptrdiff_t **operator-** (_Up *__lhs, const _Pointer_adapter &__rhs)
- template<typename _Up>
  std::ptrdiff_t **operator-** (const _Pointer_adapter &__lhs, _Up *__rhs)
- std::ptrdiff_t **operator-** (const _Pointer_adapter &__lhs, element_type *__rhs)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, int __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, long __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, long long __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, short __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, unsigned int __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, unsigned long __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, unsigned long long __offset)
- _Pointer_adapter **operator-** (const _Pointer_adapter &__lhs, unsigned short __offset)
- std::ptrdiff_t **operator-** (element_type *__lhs, const _Pointer_adapter &__rhs)
- std::strong_ordering **operator<=>** (const _Pointer_adapter &__lhs, const _Pointer_adapter &__rhs) noexcept

6.139.1 Detailed Description

template<typename _Storage_policy>
class __gnu_cxx::_Pointer_adapter< _Storage_policy >

The following provides an 'alternative pointer' that works with the containers when specified as the pointer typedef of the allocator.
The pointer type used with the containers doesn't have to be this class, but it must support the implicit conversions, pointer arithmetic, comparison operators, etc. that are supported by this class, and avoid raising compile-time ambiguities. Because creating a working pointer can be challenging, this pointer template was designed to wrapper an easier storage policy type, so that it becomes reusable for creating other pointer types.

A key point of this class is also that it allows container writers to 'assume' Allocator::pointer is a typedef for a normal pointer. This class supports most of the conventions of a true pointer, and can, for instance handle implicit conversion to const and base class pointer types. The only impositions on container writers to support extended pointers are: 1) use the Allocator::pointer typedef appropriately for pointer types. 2) if you need pointer casting, use the __pointer_cast<> functions from ext/cast.h. This allows pointer cast operations to be overloaded as necessary by custom pointers.
Note: The const qualifier works with this pointer adapter as follows:
_Tp* == _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp* == _Pointer_adapter<_Std_pointer_impl<const _Tp> >; _Tp* const == const _Pointer_adapter<_Std_pointer_impl<_Tp> >; const _Tp* const == const _Pointer_↩ adapter<_Std_pointer_impl<const _Tp> >;
The documentation for this class was generated from the following file:

- pointer.h

6.140 __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp > Class Template Reference

`#include <base.h>`

Public Types

- typedef _DifferenceTp **_DifferenceType**
- typedef _PseudoSequenceIterator< _Tp, uint64_t > **iterator**

Public Member Functions

- _PseudoSequence (const _Tp &__val, _DifferenceType __count)
- iterator begin () const
- iterator end () const

6.140.1 Detailed Description

template<typename _Tp, typename _DifferenceTp>
class __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >

Sequence that conceptually consists of multiple copies of the same element. The copies are not stored explicitly, of course.

Parameters

_Tp	Sequence _M_value type.
_DifferenceTp	Sequence difference type.

6.140.2 Constructor & Destructor Documentation

_PseudoSequence()

```
template<typename _Tp, typename _DifferenceTp>
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::_PseudoSequence (
            const _Tp & __val,
            _DifferenceType __count)  [inline]
```

Constructor.

Parameters

__val	Element of the sequence.
__count	Number of (virtual) copies.

6.140.3 Member Function Documentation

begin()

```
template<typename _Tp, typename _DifferenceTp>
iterator __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::begin () const  [inline]
```

Begin iterator.

end()

```
template<typename _Tp, typename _DifferenceTp>
iterator __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >::end () const  [inline]
```

End iterator.

The documentation for this class was generated from the following file:

- base.h

6.141 __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp > Class Template Reference

```
#include <base.h>
```

Public Types

- typedef _DifferenceTp **_DifferenceType**

Public Member Functions

- **_PseudoSequenceIterator** (const _Tp &__val, _DifferenceType __pos)
- bool **operator!=** (const _PseudoSequenceIterator &__i2)
- const _Tp & **operator*** () const
- _PseudoSequenceIterator & **operator++** ()
- _PseudoSequenceIterator **operator++** (int)
- _DifferenceType **operator-** (const _PseudoSequenceIterator &__i2)
- bool **operator==** (const _PseudoSequenceIterator &__i2)
- const _Tp & **operator[ ]** (_DifferenceType) const

6.141.1 Detailed Description

template<typename _Tp, typename _DifferenceTp>
class __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >

_Iterator associated with __gnu_parallel::_PseudoSequence. If features the usual random-access iterator functionality.

Parameters

_Tp	Sequence _M_value type.
_DifferenceTp	Sequence difference type.

The documentation for this class was generated from the following file:

- base.h

6.142 __gnu_parallel::_QSBThreadLocal< _RAIter > Struct Template Reference

`#include <balanced_quicksort.h>`

Public Types

- typedef _TraitsType::difference_type **_DifferenceType**
- typedef std::pair< _RAIter, _RAIter > _Piece
- typedef std::iterator_traits< _RAIter > **_TraitsType**

Public Member Functions

- _QSBThreadLocal (int __queue_size)

Public Attributes

- volatile _DifferenceType * _M_elements_leftover
- _Piece _M_global
- _Piece _M_initial
- _RestrictedBoundedConcurrentQueue< _Piece > _M_leftover_parts
- _ThreadIndex _M_num_threads

6.142.1 Detailed Description

template<typename _RAIter>
struct __gnu_parallel::_QSBThreadLocal< _RAIter >

Information local to one thread in the parallel quicksort run.

6.142.2 Member Typedef Documentation

_Piece

```
template<typename _RAIter>
typedef std::pair<_RAIter, _RAIter> __gnu_parallel::_QSBThreadLocal< _RAIter >::_Piece
```

Continuous part of the sequence, described by an iterator pair.

6.142.3 Constructor & Destructor Documentation

_QSBThreadLocal()

```
template<typename _RAIter>
__gnu_parallel::_QSBThreadLocal< _RAIter >::_QSBThreadLocal (
            int __queue_size)  [inline]
```

Constructor.

Parameters

__queue_size	size of the work-stealing queue.

References _M_leftover_parts.

6.142.4 Member Data Documentation

_M_elements_leftover

```
template<typename _RAIter>
volatile _DifferenceType* __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_elements_leftover
```

Pointer to a counter of elements left over to sort.
Referenced by __gnu_parallel::__qsb_conquer(), and __gnu_parallel::__qsb_local_sort_with_helping().

_M_global

```
template<typename _RAIter>
_Piece __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_global
```

The complete sequence to sort.

_M_initial

```
template<typename _RAIter>
_Piece __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_initial
```

Initial piece to work on.
Referenced by __gnu_parallel::__qsb_conquer(), and __gnu_parallel::__qsb_local_sort_with_helping().

_M_leftover_parts

```
template<typename _RAIter>
_RestrictedBoundedConcurrentQueue<_Piece> __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_↩
leftover_parts
```

Work-stealing queue.
Referenced by _QSBThreadLocal(), and __gnu_parallel::__qsb_local_sort_with_helping().

_M_num_threads

```
template<typename _RAIter>
_ThreadIndex __gnu_parallel::_QSBThreadLocal< _RAIter >::_M_num_threads
```

Number of threads involved in this algorithm.
Referenced by __gnu_parallel::__qsb_local_sort_with_helping().
The documentation for this struct was generated from the following file:

- balanced_quicksort.h

6.143 std::__detail::_Quoted_string< _String, _CharT > Struct Template Reference

```
#include <quoted_string.h>
```

Public Member Functions

- **_Quoted_string** (_String __str, _CharT __del, _CharT __esc)
- _Quoted_string & **operator=** (_Quoted_string &)=delete

Public Attributes

- _CharT **_M_delim**
- _CharT **_M_escape**
- _String **_M_string**

Friends

- template<typename _CharT2, typename _Traits, typename _String2>
  std::basic_ostream< _CharT2, _Traits > & **operator**<< (std::basic_ostream< _CharT2, _Traits > &__os, const _Quoted_string< _String2, _CharT2 > &__str)
- template<typename _CharT2, typename _Traits>
  std::basic_ostream< _CharT2, _Traits > & **operator**<< (std::basic_ostream< _CharT2, _Traits > &__os, const _Quoted_string< const _CharT2 *, _CharT2 > &__str)
- template<typename _CharT2, typename _Traits, typename _Alloc>
  std::basic_istream< _CharT2, _Traits > & **operator**>> (std::basic_istream< _CharT2, _Traits > &__is, const _Quoted_string< basic_string< _CharT2, _Traits, _Alloc > &, _CharT2 > &__str)

6.143.1 Detailed Description

template<typename _String, typename _CharT>
struct std::__detail::_Quoted_string< _String, _CharT >

Struct for delimited strings.
The documentation for this struct was generated from the following file:

- quoted_string.h

6.144 __gnu_parallel::_RandomNumber Class Reference

`#include <random_number.h>`

Public Member Functions

- _RandomNumber ()
- _RandomNumber (uint32_t __seed, uint64_t _M_supremum=0x100000000ULL)
- unsigned long __genrand_bits (int __bits)
- uint32_t operator() ()
- uint32_t operator() (uint64_t local_supremum)

6.144.1 Detailed Description

Random number generator, based on the Mersenne twister.

6.144.2 Constructor & Destructor Documentation

_RandomNumber() [1/2]

```
__gnu_parallel::_RandomNumber::_RandomNumber ()  [inline]
```

Default constructor. Seed with 0.

_RandomNumber() [2/2]

```
__gnu_parallel::_RandomNumber::_RandomNumber (
            uint32_t __seed,
            uint64_t _M_supremum = 0x100000000ULL)  [inline]
```

Constructor.

Parameters

__seed	Random __seed.

_M_supremum	Generate integer random numbers in the interval [0,_M_supremum).

6.144.3 Member Function Documentation

__genrand_bits()

```
unsigned long __gnu_parallel::_RandomNumber::__genrand_bits (
            int __bits)  [inline]
```

Generate a number of random bits, run-time parameter.

Parameters

__bits	Number of bits to generate.

operator()() [1/2]

```
uint32_t __gnu_parallel::_RandomNumber::operator() ()  [inline]
```

Generate unsigned random 32-bit integer.

operator()() [2/2]

```
uint32_t __gnu_parallel::_RandomNumber::operator() (
            uint64_t local_supremum)  [inline]
```

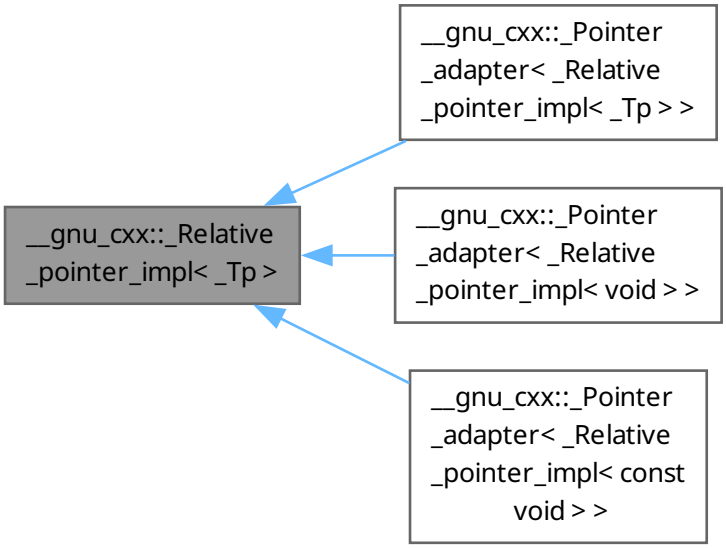
Generate unsigned random 32-bit integer in the interval [0,local_supremum).
The documentation for this class was generated from the following file:

- random_number.h

6.145 __gnu_cxx::_Relative_pointer_impl< _Tp > Class Template Reference

```
#include <pointer.h>
```

Inheritance diagram for __gnu_cxx::_Relative_pointer_impl< _Tp >:



Public Types

- typedef _Tp **element_type**

Public Member Functions

- _Tp * **get** () const
- bool **operator<** (const _Relative_pointer_impl &__rarg) const
- bool **operator==** (const _Relative_pointer_impl &__rarg) const
- void **set** (_Tp *__arg)

6.145.1 Detailed Description

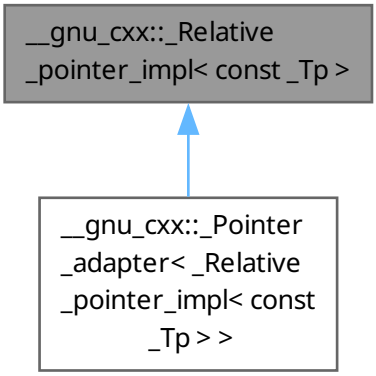
template<typename _Tp>
class __gnu_cxx::_Relative_pointer_impl< _Tp >

A storage policy for use with _Pointer_adapter<> which stores the pointer's address as an offset value which is relative to its own address.
This is intended for pointers within shared memory regions which might be mapped at different addresses by different processes. For null pointers, a value of 1 is used. (0 is legitimate sometimes for nodes in circularly linked lists) This value was chosen as the least likely to generate an incorrect null, As there is no reason why any normal pointer would point 1 byte into its own pointer address.
The documentation for this class was generated from the following file:

- pointer.h

6.146 __gnu_cxx::_Relative_pointer_impl< const _Tp > Class Template Reference

`#include <pointer.h>`
Inheritance diagram for __gnu_cxx::_Relative_pointer_impl< const _Tp >:



Public Types

- typedef const _Tp **element_type**

Public Member Functions

- const _Tp * **get** () const
- bool **operator<** (const _Relative_pointer_impl &__rarg) const
- bool **operator==** (const _Relative_pointer_impl &__rarg) const
- void **set** (const _Tp *__arg)

6.146.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::_Relative_pointer_impl< const _Tp >

Relative_pointer_impl needs a specialization for const T because of the casting done during pointer arithmetic.
The documentation for this class was generated from the following file:

- pointer.h

6.147 __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp > Class Template Reference

`#include <queue.h>`

Public Member Functions

- _RestrictedBoundedConcurrentQueue (_SequenceIndex __max_size)
- ~_RestrictedBoundedConcurrentQueue ()

- bool pop_back (_Tp &__t)
- bool pop_front (_Tp &__t)
- void push_front (const _Tp &__t)

6.147.1 Detailed Description

template<typename _Tp>
class __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >

Double-ended queue of bounded size, allowing lock-free atomic access. push_front() and pop_front() must not be called concurrently to each other, while pop_back() can be called concurrently at all times. `empty()`, `size()`, and `top()` are intentionally not provided. Calling them would not make sense in a concurrent setting.

Parameters

_Tp	Contained element type.

6.147.2 Constructor & Destructor Documentation

_RestrictedBoundedConcurrentQueue()

```
template<typename _Tp>
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::_RestrictedBoundedConcurrentQueue (
            _SequenceIndex __max_size)  [inline]
```

Constructor. Not to be called concurrent, of course.

Parameters

__max_size	Maximal number of elements to be contained.

References __gnu_parallel::__encode2().

~_RestrictedBoundedConcurrentQueue()

```
template<typename _Tp>
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::~_RestrictedBoundedConcurrentQueue ()
[inline]
```

Destructor. Not to be called concurrent, of course.

6.147.3 Member Function Documentation

pop_back()

```
template<typename _Tp>
bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_back (
            _Tp & __t)  [inline]
```

Pops one element from the queue at the front end. Must not be called concurrently with pop_front().
References __gnu_parallel::__compare_and_swap(), __gnu_parallel::__decode2(), and __gnu_parallel::__encode2().

pop_front()

```
template<typename _Tp>
bool __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::pop_front (
            _Tp & __t)  [inline]
```

Pops one element from the queue at the front end. Must not be called concurrently with pop_front().
References __gnu_parallel::__compare_and_swap(), __gnu_parallel::__decode2(), and __gnu_parallel::__encode2().

push_front()

```
template<typename _Tp>
void __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >::push_front (
            const _Tp & __t)  [inline]
```

Pushes one element into the queue at the front end. Must not be called concurrently with pop_front().
References __gnu_parallel::__decode2(), __gnu_parallel::__encode2(), and __gnu_parallel::__fetch_and_add().
The documentation for this class was generated from the following file:

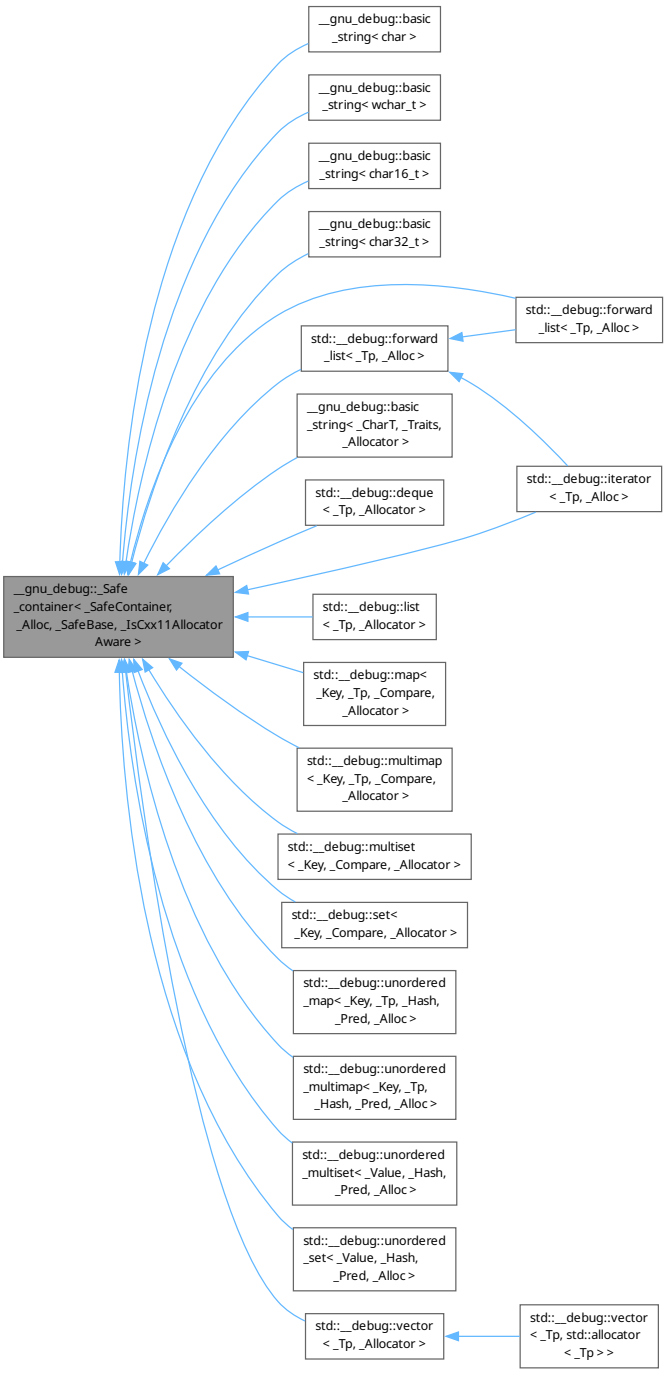
- queue.h

6.148 __gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware > Class Template Reference

```
#include <safe_container.h>
```

Inheritance diagram for __gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware

>:



Protected Member Functions

- **_Safe_container** (_Safe_container &&)=default
- constexpr **_Safe_container** (_Safe_container &&__x, const _Alloc &__a)
- **_Safe_container** (const _Safe_container &)=default

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept
- constexpr _Safe_container & **operator=** (_Safe_container &&__x) noexcept
- constexpr _Safe_container & **operator=** (const _Safe_container &) noexcept=default

6.148.1 Detailed Description

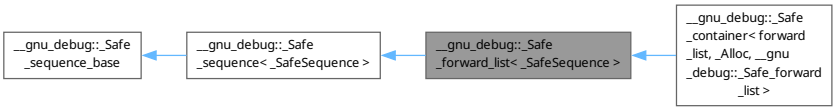
template<typename _SafeContainer, typename _Alloc, template< typename > class _SafeBase, bool _IsCxx11AllocatorAware = true>
class __gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware >

Safe class dealing with some allocator dependent operations.
The documentation for this class was generated from the following file:

- safe_container.h

6.149 __gnu_debug::_Safe_forward_list< _SafeSequence > Class Template Reference

`#include <forward_list>`
Inheritance diagram for __gnu_debug::_Safe_forward_list< _SafeSequence >:



Public Member Functions

- constexpr void _M_invalidate_if (_Predicate __pred) const
- void _M_transfer_from_if (const _Safe_sequence &__from, _Predicate __pred) const

Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all () const
- void _M_detach_singular () const
- __gnu_cxx::__mutex & _M_get_mutex () const noexcept
- void **_M_invalidate_all** () const
- void _M_revalidate_singular () const
- void **_M_swap** (const _Safe_forward_list &) const noexcept
- void _M_swap (const _Safe_sequence_base &__x) const noexcept

6.149.1 Detailed Description

template<typename _SafeSequence>
class __gnu_debug::_Safe_forward_list< _SafeSequence >

Special iterators swap and invalidation for forward_list because of the before_begin iterator.

6.149.2 Member Function Documentation

_M_detach_all()

void __gnu_debug::_Safe_sequence_base::_M_detach_all () const [protected], [inherited]
Detach all iterators, leaving them singular.

_M_detach_singular()

void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const [protected], [inherited]
Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

_M_get_mutex()

__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const [protected], [noexcept], [inherited]
For use in _Safe_sequence.

_M_invalidate_if()

void __gnu_debug::_Safe_sequence< _SafeSequence >::_M_invalidate_if (
_Predicate __pred) const [inline], [constexpr], [inherited]
Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

_M_revalidate_singular()

void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const [protected], [inherited]
Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

_M_swap()

void __gnu_debug::_Safe_sequence_base::_M_swap (
const _Safe_sequence_base & __x) const [protected], [noexcept], [inherited]
Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

_M_transfer_from_if()

void __gnu_debug::_Safe_sequence< _SafeSequence >::_M_transfer_from_if (
const _Safe_sequence< _SafeSequence > & __from,
_Predicate __pred) const [inherited]
Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.
References _M_iterators.

6.149.3 Member Data Documentation

_M_const_iterators

_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators [mutable], [inherited]
The list of constant iterators that reference this container.

_M_iterators

_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators [mutable], [inherited]
The list of mutable iterators that reference this container.
Referenced by _M_transfer_from_if().

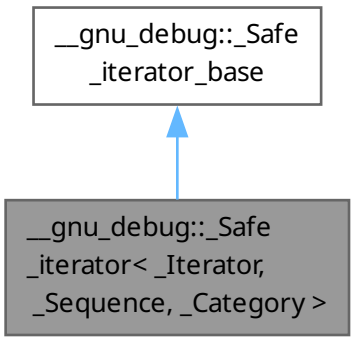
_M_version

unsigned int __gnu_debug::_Safe_sequence_base::_M_version [mutable], [inherited]
The container version number. This number may never be 0.
The documentation for this class was generated from the following file:

- debug/forward_list

6.150 __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category > Class Template Reference

#include <safe_iterator.h>
Inheritance diagram for __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >:



Public Types

- typedef _Safe_iterator< _Iterator, _Sequence, iterator_category > **_Self**
- typedef _Traits::difference_type **difference_type**
- typedef _Traits::iterator_category **iterator_category**
- using **iterator_concept**
- typedef _Iterator **iterator_type**
- typedef _Traits::pointer **pointer**
- typedef _Traits::reference **reference**
- typedef _Traits::value_type **value_type**

Public Member Functions

- constexpr _Safe_iterator () noexcept
- constexpr _Safe_iterator (_Iterator __i, const _Safe_sequence_base *__seq) noexcept
- constexpr _Safe_iterator (_Safe_iterator &&__x) noexcept

- constexpr _Safe_iterator (const _Safe_iterator &__x) noexcept
- template<typename _MutableIterator>
  constexpr _Safe_iterator (const _Safe_iterator< _MutableIterator, _Sequence, typename __gnu_cxx::__enable_if< _IsConstant::__value &&std::__are_same< _MutableIterator, _OtherIterator >::__value, _Category >::_type > &__x) noexcept
- void _M_attach (const _Safe_sequence_base *__seq)
- void _M_attach_single (const _Safe_sequence_base *__seq)
- bool _M_attached_to (const _Safe_sequence_base *__seq) const
- bool _M_before_dereferenceable () const
- template<typename _Diff>
  bool **_M_can_advance** (const std::pair< _Diff, _Distance_precision > &__dist, int __way) const
- bool **_M_can_advance** (difference_type __n, bool __strict=false) const
- bool _M_can_compare (const _Safe_iterator_base &__x) const noexcept
- bool _M_dereferenceable () const
- void _M_detach_single () noexcept
- _Distance_traits< _Iterator >::__type **_M_get_distance_from_begin** () const
- _Distance_traits< _Iterator >::__type **_M_get_distance_to** (const _Safe_iterator &__rhs) const
- _Distance_traits< _Iterator >::__type **_M_get_distance_to_end** () const
- __gnu_cxx::__conditional_type< _IsConstant::__value, const_Sequence *, _Sequence * >::__type **_M_get_sequence** () const
- bool _M_incrementable () const
- void _M_invalidate ()
- bool _M_is_before_begin () const
- constexpr bool _M_is_begin () const
- bool _M_is_beginnest () const
- bool _M_is_end () const
- void _M_reset () noexcept
- bool _M_singular () const noexcept
- void _M_unlink () noexcept
- bool **_M_valid_range** (const _Safe_iterator &__rhs, std::pair< difference_type, _Distance_precision > &__dist, bool __check_dereferenceable=true) const
- bool _M_value_initialized () const
- constexpr const _Iterator & **base** () const noexcept
- constexpr _Iterator & base () noexcept
- constexpr operator _Iterator () const noexcept
- constexpr reference operator* () const noexcept
- constexpr _Safe_iterator & operator++ () noexcept
- constexpr _Safe_iterator operator++ (int) noexcept
- constexpr pointer operator-> () const noexcept
- constexpr _Safe_iterator & operator= (_Safe_iterator &&__x) noexcept
- constexpr _Safe_iterator & operator= (const _Safe_iterator &__x) noexcept

Static Public Member Functions

- static constexpr bool _S_constant ()
- template<typename _Fn>
  static constexpr _Self **_S_for_each_segment** (_Self __first, _Self __last, _Fn __func)

Public Attributes

- _Safe_iterator_base * _M_next
- _Safe_iterator_base * _M_prior
- const _Safe_sequence_base * _M_sequence
- unsigned int _M_version

Static Public Attributes

- static const bool **_S_enable_for_each_segment**

Protected Types

- typedef std::__are_same< typename _Sequence::_Base::const_iterator, _Iterator > **_IsConstant**
- typedef __gnu_cxx::__conditional_type< _IsConstant::__value, typename_Sequence::_Base::iterator, typename↩ _Sequence::_Base::const_iterator >::__type **_OtherIterator**

Protected Member Functions

- constexpr **_Safe_iterator** (const _Safe_iterator &__x, _Unchecked) noexcept
- void _M_attach (const _Safe_sequence_base *__seq, bool __constant)
- void _M_attach_single (const _Safe_sequence_base *__seq, bool __constant) noexcept
- void _M_detach ()
- __gnu_cxx::__mutex & _M_get_mutex () noexcept

Friends

- template<typename _IteR>
  constexpr friend bool **operator==** (const _Self &__lhs, const _Safe_iterator< _IteR, _Sequence, iterator_category > &__rhs) noexcept
- constexpr friend bool **operator==** (const _Self &__lhs, const _Self &__rhs) noexcept

6.150.1 Detailed Description

template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_traits<_↩ Iterator>::iterator_category>
class __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >

Safe iterator wrapper.
The class template _Safe_iterator is a wrapper around an iterator that tracks the iterator's movement among sequences and checks that operations performed on the "safe" iterator are legal. In additional to the basic iterator operations (which are validated, and then passed to the underlying iterator), _Safe_iterator has member functions for iterator invalidation, attaching/detaching the iterator from sequences, and querying the iterator's state.
Note that _Iterator must be the first base class so that it gets initialized before the iterator is being attached to the container's list of iterators and it is being detached before _Iterator get destroyed. Otherwise it would result in a data race.

6.150.2 Constructor & Destructor Documentation

_Safe_iterator() [1/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator ()  [inline],
[constexpr], [noexcept]
```

Postcondition

the iterator is singular and unattached

_Safe_iterator() [2/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
            _Iterator __i,
            const _Safe_sequence_base * __seq)  [inline], [constexpr], [noexcept]
```

Safe iterator construction from an unsafe iterator and its sequence.

Precondition

`seq` is not NULL

Postcondition

this is not singular

_Safe_iterator() [3/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
            const _Safe_iterator< _Iterator, _Sequence, _Category > & __x)  [inline], [constexpr],
[noexcept]
```

Copy construction.

_Safe_iterator() [4/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
            _Safe_iterator< _Iterator, _Sequence, _Category > && __x)  [inline], [constexpr],
[noexcept]
```

Move construction.

Postcondition

__x is singular and unattached

_Safe_iterator() [5/5]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
template<typename _MutableIterator>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_Safe_iterator (
            const _Safe_iterator< _MutableIterator, _Sequence, typename __gnu_cxx::__enable_if<
_IsConstant::__value &&std::__are_same< _MutableIterator, _OtherIterator >::__value, _Category
>::__type > & __x)  [inline], [constexpr], [noexcept]
```

Converting constructor from a mutable iterator to a constant iterator.

6.150.3 Member Function Documentation

_M_attach() [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
```

```
void __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_attach (
            const _Safe_sequence_base * __seq)  [inline]
```

Attach iterator to the given sequence.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator
and __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=().

_M_attach() [2/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.
References _M_attach().
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach(), and _M_attach().

_M_attach_single() [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
void __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_attach_single (
            const _Safe_sequence_base * __seq)  [inline]
```

Likewise, but not thread-safe.

_M_attach_single() [2/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.
References _M_attach_single().
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach_single(), and _M_attach_single().

_M_attached_to()

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
            const _Safe_sequence_base * __seq) const  [inline], [inherited]
```

Determines if we are attached to the given sequence.
References _M_attached_to(), and _M_sequence.
Referenced by _M_attached_to().

_M_before_dereferenceable()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_before_dereferenceable ()
const  [inline]
```

Is the iterator before a dereferenceable one?

_M_can_compare()

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
            const _Safe_iterator_base & __x) const  [noexcept], [inherited]
```

Can we compare this iterator to the given iterator ___x? Returns true if both iterators are nonsingular and reference the same sequence.
References _Safe_iterator_base(), and _M_can_compare().
Referenced by _M_can_compare().

_M_dereferenceable()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_dereferenceable () const
[inline]
```

Is the iterator dereferenceable?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator*(), and __gnu_debug::_Safe_iterator< _Base_iterator, m

_M_detach()

```
void __gnu_debug::_Safe_iterator_base::_M_detach ()  [protected], [inherited]
```

Detach the iterator for whatever sequence it is attached to, if any.
References _M_detach().
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), _M_detach(), __gnu_debug::_Safe_iterator< _Ba
and __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=().

_M_detach_single()

```
void __gnu_debug::_Safe_iterator_base::_M_detach_single ()  [noexcept], [inherited]
```

Likewise, but not thread-safe.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex ()  [protected], [noexcept],
[inherited]
```

For use in _Safe_iterator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++(), __gnu_debug::_Safe_local_iterator< _OtherIterator,
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_incrementable()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_incrementable () const
[inline]
```

Is the iterator incrementable?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_before_dereferenceable(), __gnu_debug::_Safe_iterator< _Ba
and __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++().

_M_invalidate()

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate ()  [inline], [inherited]
```

Invalidate the iterator, making it singular.
References _M_invalidate(), and _M_version.
Referenced by _M_invalidate().

_M_is_before_begin()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_before_begin () const
[inline]
```

Is this iterator equal to the sequence's before_begin() iterator if any?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable().

_M_is_begin()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_begin () const  [inline],
[constexpr]
```

Is this iterator equal to the sequence's begin() iterator?

_M_is_beginnest()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_beginnest () const
[inline]
```

Is this iterator equal to the sequence's before_begin() iterator if any or begin() otherwise?

_M_is_end()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_is_end () const  [inline]
```

Is this iterator equal to the sequence's end() iterator?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable(), and __gnu_debug::_Safe_iterator< _Base

_M_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset ()  [noexcept], [inherited]
```

Reset all member variables

_M_singular()

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const  [noexcept], [inherited]
```

Is this iterator singular?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContaine
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Other
__gnu_debug::__check_singular_aux(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_dereferenceable(), __gnu_debug::_Safe_iterator< _Base_itera
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Base_iterator
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_unlink()

```
void __gnu_debug::_Safe_iterator_base::_M_unlink ()  [inline], [noexcept], [inherited]
```

Unlink itself

References _M_next, _M_prior, and _M_unlink().
Referenced by _M_unlink().

_M_value_initialized()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_M_value_initialized ()
const  [inline]
```

Is the iterator value-initialized?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator
and __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=().

_S_constant()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
constexpr bool __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::_S_constant ()
[inline], [static], [constexpr]
```

Determine if this is a constant iterator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach(), and __gnu_debug::_Safe_iterator< _Base_iterator, r

base()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
_Iterator & __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::base ()  [inline],
[constexpr], [noexcept]
```

Return the underlying iterator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_is_begin(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_is_en
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_value_initialized(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::op
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator->
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), and __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator

operator _Iterator()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator _Iterator () const  [inline],
[constexpr], [noexcept]
```

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

operator*()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
reference __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator* () const
[inline], [nodiscard], [constexpr], [noexcept]
```

Iterator dereference.

Precondition

iterator is dereferenceable

operator++() [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
_Safe_iterator & __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator++ ()
[inline], [constexpr], [noexcept]
```

Iterator preincrement.

Precondition

iterator is incrementable

operator++() [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
_Safe_iterator __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator++ (
            int )  [inline], [constexpr], [noexcept]
```

Iterator postincrement.

Precondition

iterator is incrementable

operator->()

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
pointer __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator-> () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Iterator dereference.

Precondition

iterator is dereferenceable

operator=() [1/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
_Safe_iterator & __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator= (
            _Safe_iterator< _Iterator, _Sequence, _Category > && __x)  [inline], [constexpr],
[noexcept]
```

Move assignment.

Postcondition

__x is singular and unattached

operator=() [2/2]

```
template<typename _Iterator, typename _Sequence, typename _Category = typename std::iterator_↩
traits<_Iterator>::iterator_category>
_Safe_iterator & __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >::operator= (
            const _Safe_iterator< _Iterator, _Sequence, _Category > & __x)  [inline], [constexpr],
[noexcept]
```

Copy assignment.

6.150.4 Member Data Documentation

_M_next

_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]

Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _Sequence >::_M and _M_unlink().

_M_prior

_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]

Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _Sequence >::_M and _M_unlink().

_M_sequence

const _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]

The sequence this iterator references; may be NULL to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.

Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_attached_to(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_version

unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_invalidate(), and __gnu_debug::_Safe_sequence

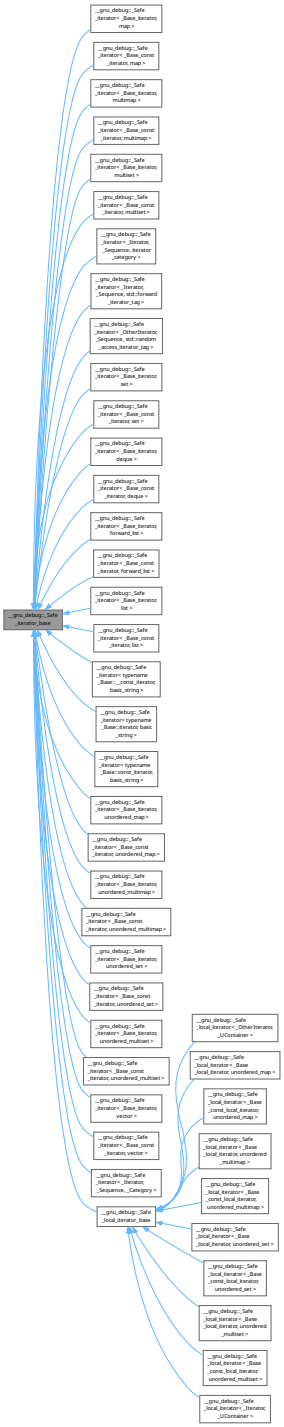
The documentation for this class was generated from the following files:

- boost_concept_check.h
- safe_iterator.h
- safe_iterator.tcc

6.151 __gnu_debug::_Safe_iterator_base Class Reference

#include <safe_base.h>

Inheritance diagram for __gnu_debug::_Safe_iterator_base:



Public Member Functions

- bool _M_attached_to (const _Safe_sequence_base *__seq) const
- bool _M_can_compare (const _Safe_iterator_base &__x) const noexcept

- void _M_detach_single () noexcept
- void _M_invalidate ()
- void _M_reset () noexcept
- bool _M_singular () const noexcept
- void _M_unlink () noexcept

Public Attributes

- _Safe_iterator_base * _M_next
- _Safe_iterator_base * _M_prior
- const _Safe_sequence_base * _M_sequence
- unsigned int _M_version

Protected Member Functions

- constexpr _Safe_iterator_base ()
- constexpr _Safe_iterator_base (const _Safe_iterator_base &__x, bool __constant)
- constexpr _Safe_iterator_base (const _Safe_sequence_base *__seq, bool __constant)
- void _M_attach (const _Safe_sequence_base *__seq, bool __constant)
- void _M_attach_single (const _Safe_sequence_base *__seq, bool __constant) noexcept
- void _M_detach ()
- __gnu_cxx::__mutex & _M_get_mutex () noexcept

Friends

- class **_Safe_sequence_base**

6.151.1 Detailed Description

Basic functionality for a *safe* iterator.
The _Safe_iterator_base base class implements the functionality of a safe iterator that is not specific to a particular iterator type. It contains a pointer back to the sequence it references along with iterator version information and pointers to form a doubly-linked list of iterators referenced by the container.
This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

6.151.2 Constructor & Destructor Documentation

_Safe_iterator_base() [1/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base ()  [inline], [constexpr], [protected]
```

Initializes the iterator and makes it singular.
References _M_next, _M_prior, _M_sequence, and _M_version.
Referenced by _Safe_iterator_base(), and _M_can_compare().

_Safe_iterator_base() [2/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (
            const _Safe_sequence_base * __seq,
            bool __constant)  [inline], [constexpr], [protected]
```

Initialize the iterator to reference the sequence pointed to by __seq. __constant is true when we are initializing a constant iterator, and false if it is a mutable iterator. Note that __seq may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference __seq and be nonsingular.
References _M_attach(), _M_next, _M_prior, _M_sequence, and _M_version.

_Safe_iterator_base() [3/3]

```
__gnu_debug::_Safe_iterator_base::_Safe_iterator_base (
            const _Safe_iterator_base & __x,
            bool __constant)  [inline], [constexpr], [protected]
```

Initializes the iterator to reference the same sequence that __x does. __constant is true if this is a constant iterator, and false if it is mutable.
References _Safe_iterator_base(), _M_attach(), _M_next, _M_prior, _M_sequence, and _M_version.

6.151.3 Member Function Documentation

_M_attach()

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.
References _M_attach().
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach(), and _M_attach().

_M_attach_single()

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [noexcept]
```

Likewise, but not thread-safe.
References _M_attach_single().
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach_single(), and _M_attach_single().

_M_attached_to()

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
            const _Safe_sequence_base * __seq) const  [inline]
```

Determines if we are attached to the given sequence.
References _M_attached_to(), and _M_sequence.
Referenced by _M_attached_to().

_M_can_compare()

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
            const _Safe_iterator_base & __x) const  [noexcept]
```

Can we compare this iterator to the given iterator __x? Returns true if both iterators are nonsingular and reference the same sequence.
References _Safe_iterator_base(), and _M_can_compare().
Referenced by _M_can_compare().

_M_detach()

```
void __gnu_debug::_Safe_iterator_base::_M_detach ()  [protected]
```

Detach the iterator for whatever sequence it is attached to, if any.
References _M_detach().
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), _M_detach(), __gnu_debug::_Safe_iterator< _Ba
and __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=().

_M_detach_single()

```
void __gnu_debug::_Safe_iterator_base::_M_detach_single ()  [noexcept]
```
Likewise, but not thread-safe.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex ()  [protected], [noexcept]
```
For use in _Safe_iterator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++(), __gnu_debug::_Safe_local_iterator< _OtherIterator,
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_invalidate()

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate ()  [inline]
```
Invalidate the iterator, making it singular.
References _M_invalidate(), and _M_version.
Referenced by _M_invalidate().

_M_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset ()  [noexcept]
```
Reset all member variables

_M_singular()

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const  [noexcept]
```
Is this iterator singular?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContaine
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Other
__gnu_debug::__check_singular_aux(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_dereferenceable(), __gnu_debug::_Safe_iterator< _Base_itera
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Base_iterato
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_unlink()

```
void __gnu_debug::_Safe_iterator_base::_M_unlink ()  [inline], [noexcept]
```
Unlink itself
References _M_next, _M_prior, and _M_unlink().
Referenced by _M_unlink().

6.151.4 Member Data Documentation

_M_next

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next
```
Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _Sequence >::_M
and _M_unlink().

_M_prior

_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior

Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _Sequence >::_M and _M_unlink().

_M_sequence

const _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence

The sequence this iterator references; may be NULL to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.

Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_attached_to(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_version

unsigned int __gnu_debug::_Safe_iterator_base::_M_version

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_invalidate(), and __gnu_debug::_Safe_sequence

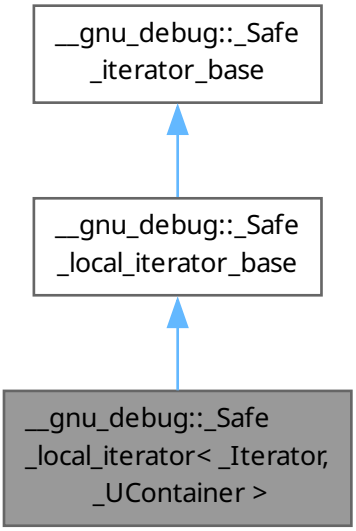
The documentation for this class was generated from the following file:

- safe_base.h

6.152 __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer > Class Template Reference

#include <safe_local_iterator.h>

Inheritance diagram for __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >:



Public Types

- typedef _Traits::difference_type **difference_type**
- typedef _Traits::iterator_category **iterator_category**
- typedef _Iterator **iterator_type**
- typedef _Traits::pointer **pointer**
- typedef _Traits::reference **reference**
- typedef _Traits::value_type **value_type**

Public Member Functions

- _Safe_local_iterator () noexcept
- _Safe_local_iterator (_Iterator __i, const _Safe_unordered_container_base *__cont)
- _Safe_local_iterator (_Safe_local_iterator &&__x) noexcept
- _Safe_local_iterator (const _Safe_local_iterator &__x) noexcept
- template<typename _MutableIterator>
  _Safe_local_iterator (const _Safe_local_iterator< _MutableIterator, typename __gnu_cxx::__enable_if< _IsConstant::__value &&std::__are_same< _MutableIterator, _OtherIterator >::__value, _UContainer >::__type > &__x) noexcept
- void _M_attach (const _Safe_unordered_container_base *__cont)
- void _M_attach_single (const _Safe_unordered_container_base *__cont)
- bool _M_attached_to (const _Safe_sequence_base *__seq) const
- bool _M_can_compare (const _Safe_iterator_base &__x) const noexcept
- bool _M_dereferenceable () const
- _Distance_traits< _Iterator >::__type **_M_get_distance_to** (const _Safe_local_iterator &__rhs) const
- std::__conditional_t< _IsConstant::__value, const _UContainer *, _UContainer * > **_M_get_ucontainer** () const

- template<typename _Other>
  bool _M_in_same_bucket (const _Safe_local_iterator< _Other, _UContainer > &__other) const
- bool _M_incrementable () const
- void _M_invalidate ()
- bool _M_is_begin () const
- bool _M_is_end () const
- void _M_reset () noexcept
- const _Safe_unordered_container_base * **_M_safe_container** () const noexcept
- bool _M_singular () const noexcept
- void _M_unlink () noexcept
- bool **_M_valid_range** (const _Safe_local_iterator &__rhs, std::pair< difference_type, _Distance_precision > &↩ __dist_info) const
- bool _M_value_initialized () const
- const _Iterator & **base** () const noexcept
- _Iterator & base () noexcept
- size_type bucket () const
- operator _Iterator () const
- reference operator* () const
- _Safe_local_iterator & operator++ ()
- _Safe_local_iterator operator++ (int)
- pointer operator-> () const
- _Safe_local_iterator & operator= (_Safe_local_iterator &&__x) noexcept
- _Safe_local_iterator & operator= (const _Safe_local_iterator &__x)

Static Public Member Functions

- static constexpr bool _S_constant ()

Public Attributes

- _Safe_iterator_base * _M_next
- _Safe_iterator_base * _M_prior
- const _Safe_sequence_base * _M_sequence
- unsigned int _M_version

Protected Member Functions

- void _M_attach (const _Safe_sequence_base *__seq, bool __constant)
- void _M_attach (const _Safe_unordered_container_base *__cont, bool __constant)
- void _M_attach_single (const _Safe_sequence_base *__seq, bool __constant) noexcept
- void _M_attach_single (const _Safe_unordered_container_base *__cont, bool __constant) noexcept
- void _M_detach ()
- void _M_detach_single () noexcept
- __gnu_cxx::__mutex & _M_get_mutex () noexcept

Friends

- bool **operator!=** (const _Self &__lhs, const _OtherSelf &__rhs) noexcept
- bool **operator!=** (const _Self &__lhs, const _Self &__rhs) noexcept
- bool **operator==** (const _Self &__lhs, const _OtherSelf &__rhs) noexcept
- bool **operator==** (const _Self &__lhs, const _Self &__rhs) noexcept

6.152.1 Detailed Description

template<typename _Iterator, typename _UContainer>
class __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >

Safe iterator wrapper.
The class template _Safe_local_iterator is a wrapper around an iterator that tracks the iterator's movement among unordered containers and checks that operations performed on the "safe" iterator are legal. In additional to the basic iterator operations (which are validated, and then passed to the underlying iterator), _Safe_local_iterator has member functions for iterator invalidation, attaching/detaching the iterator from unordered containers, and querying the iterator's state.

6.152.2 Constructor & Destructor Documentation

_Safe_local_iterator() [1/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator ()  [inline],
[noexcept]
```

Postcondition

the iterator is singular and unattached

_Safe_local_iterator() [2/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator (
            _Iterator __i,
            const _Safe_unordered_container_base * __cont)  [inline]
```

Safe iterator construction from an unsafe iterator and its unordered container.

Precondition

`cont` is not NULL

Postcondition

this is not singular

_Safe_local_iterator() [3/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator (
            const _Safe_local_iterator< _Iterator, _UContainer > & __x)  [inline], [noexcept]
```

Copy construction.

_Safe_local_iterator() [4/5]

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator (
            _Safe_local_iterator< _Iterator, _UContainer > && __x)  [inline], [noexcept]
```

Move construction.

Postcondition

__x is singular and unattached

_Safe_local_iterator() [5/5]

```
template<typename _Iterator, typename _UContainer>
template<typename _MutableIterator>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_Safe_local_iterator (
            const _Safe_local_iterator< _MutableIterator, typename __gnu_cxx::__enable_if< _↩
IsConstant::__value &&std::__are_same< _MutableIterator, _OtherIterator >::__value, _UContainer
>::__type > & __x)  [inline], [noexcept]
```

Converting constructor from a mutable iterator to a constant iterator.

6.152.3 Member Function Documentation

_M_attach() [1/3]

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.

References _M_attach().

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach(), and _M_attach().

_M_attach() [2/3]

```
template<typename _Iterator, typename _UContainer>
void __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_attach (
            const _Safe_unordered_container_base * __cont)  [inline]
```

Attach iterator to the given unordered container.

Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_it
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Other
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_attach() [3/3]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach (
            const _Safe_unordered_container_base * __cont,
            bool __constant)  [protected], [inherited]
```

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.

Referenced by _Safe_local_iterator_base(), _Safe_local_iterator_base(), and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UCon

_M_attach_single() [1/3]

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

References _M_attach_single().

Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach_single(), and _M_attach_single().

_M_attach_single() [2/3]

```
template<typename _Iterator, typename _UContainer>
void __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_attach_single (
            const _Safe_unordered_container_base * __cont)  [inline]
```

Likewise, but not thread-safe.

_M_attach_single() [3/3]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (
            const _Safe_unordered_container_base * __cont,
            bool __constant)  [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_attach_single().

_M_attached_to()

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
            const _Safe_sequence_base * __seq) const  [inline], [inherited]
```

Determines if we are attached to the given sequence.
References _M_attached_to(), and _M_sequence.
Referenced by _M_attached_to().

_M_can_compare()

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
            const _Safe_iterator_base & __x) const  [noexcept], [inherited]
```

Can we compare this iterator to the given iterator __x? Returns true if both iterators are nonsingular and reference the same sequence.
References _Safe_iterator_base(), and _M_can_compare().
Referenced by _M_can_compare().

_M_dereferenceable()

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_dereferenceable () const
[inline]
```

Is the iterator dereferenceable?
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator*(), and __gnu_debug::_Safe_local_iterator<

_M_detach()

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach ()  [protected], [inherited]
```

Detach the iterator for whatever container it is attached to, if any.
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_it
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_detach_single()

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach_single ()  [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex ()  [protected], [noexcept],
[inherited]
```

For use in _Safe_iterator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++(), __gnu_debug::_Safe_local_iterator< _OtherIterator,
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_in_same_bucket()

```
template<typename _Iterator, typename _UContainer>
template<typename _Other>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_in_same_bucket (
             const _Safe_local_iterator< _Other, _UContainer > & __other) const  [inline]
```

Is this iterator part of the same bucket as the other one?

_M_incrementable()

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_incrementable () const  [inline]
```

Is the iterator incrementable?
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator++(), and __gnu_debug::_Safe_local_iterato

_M_invalidate()

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate ()  [inline], [inherited]
```

Invalidate the iterator, making it singular.
References _M_invalidate(), and _M_version.
Referenced by _M_invalidate().

_M_is_begin()

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_is_begin () const  [inline]
```

Is this iterator equal to the container's begin(bucket) iterator?

_M_is_end()

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_is_end () const  [inline]
```

Is this iterator equal to the container's end(bucket) iterator?
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_dereferenceable(), and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_incrementable().

_M_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset ()  [noexcept], [inherited]
```

Reset all member variables

_M_singular()

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const  [noexcept], [inherited]
```

Is this iterator singular?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContaine
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Other
__gnu_debug::__check_singular_aux(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_dereferenceable(), __gnu_debug::_Safe_iterator< _Base_itera
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Base_iterator
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_unlink()

```
void __gnu_debug::_Safe_iterator_base::_M_unlink ()  [inline], [noexcept], [inherited]
```

Unlink itself

References _M_next, _M_prior, and _M_unlink().

Referenced by _M_unlink().

_M_value_initialized()

```
template<typename _Iterator, typename _UContainer>
bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_M_value_initialized () const
[inline]
```

Is the iterator value-initialized?

Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_it
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Other
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_S_constant()

```
template<typename _Iterator, typename _UContainer>
constexpr bool __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::_S_constant ()  [inline],
[static], [constexpr]
```

Determine if this is a constant iterator.

Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_attach(), and __gnu_debug::_Safe_local_iterator

base()

```
template<typename _Iterator, typename _UContainer>
_Iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::base ()  [inline], [noexcept]
```

Return the underlying iterator.

Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_it
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_is_end(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::bucket(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UCo
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator++(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

bucket()

```
template<typename _Iterator, typename _UContainer>
size_type __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::bucket () const  [inline]
```

Return the bucket.

Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_in_same_bucket(), __gnu_debug::_Safe_local_it
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_is_end().

operator _Iterator()

```
template<typename _Iterator, typename _UContainer>
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator _Iterator () const  [inline]
```

Conversion to underlying non-debug iterator to allow better interaction with non-debug containers.

operator*()

```
template<typename _Iterator, typename _UContainer>
reference __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator* () const  [inline]
```

Iterator dereference.

Precondition

iterator is dereferenceable

operator++() [1/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator++ ()
[inline]
```

Iterator preincrement.

Precondition

iterator is incrementable

operator++() [2/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator++ (
            int )  [inline]
```

Iterator postincrement.

Precondition

iterator is incrementable

operator->()

```
template<typename _Iterator, typename _UContainer>
pointer __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator-> () const  [inline]
```

Iterator dereference.

Precondition

iterator is dereferenceable

operator=() [1/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator= (
            _Safe_local_iterator< _Iterator, _UContainer > && __x)  [inline], [noexcept]
```

Move assignment.

Postcondition

__x is singular and unattached

operator=() [2/2]

```
template<typename _Iterator, typename _UContainer>
_Safe_local_iterator & __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >::operator= (
            const _Safe_local_iterator< _Iterator, _UContainer > & __x)  [inline]
```

Copy assignment.

6.152.4 Member Data Documentation

_M_next

_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]
Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _Sequence >::_M
and _M_unlink().

_M_prior

_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]
Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _Sequence >::_M
and _M_unlink().

_M_sequence

const _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]
The sequence this iterator references; may be NULL to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), _Safe_iterator_base(), _Safe_iterator_base(),
_Safe_iterator_base(), _M_attached_to(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(),
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_version

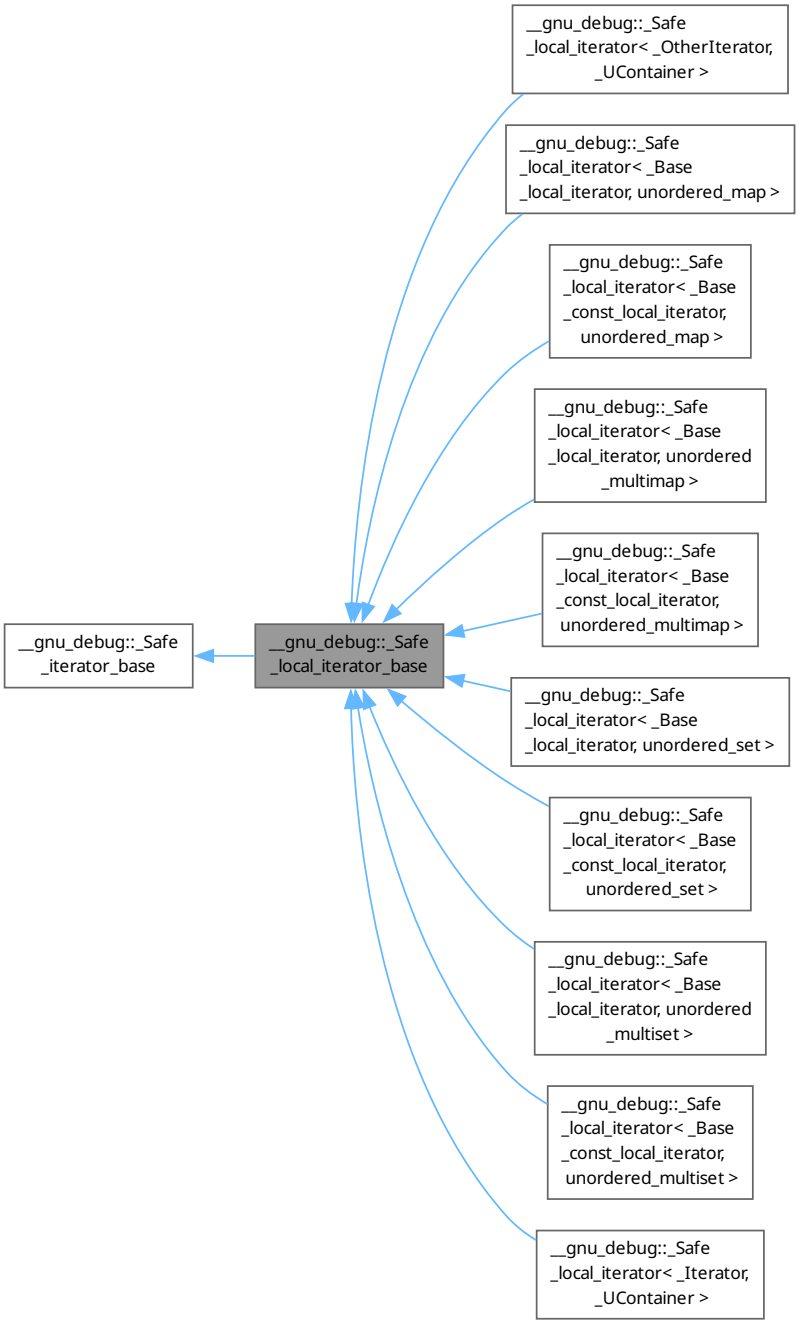
unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]
The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_invalidate(), and __gnu_debug::_Safe_sequence
The documentation for this class was generated from the following files:

- formatter.h
- safe_local_iterator.h
- safe_local_iterator.tcc

6.153 __gnu_debug::_Safe_local_iterator_base Class Reference

#include <safe_unordered_base.h>

Inheritance diagram for __gnu_debug::_Safe_local_iterator_base:



Public Member Functions

- bool _M_attached_to (const _Safe_sequence_base *__seq) const
- bool _M_can_compare (const _Safe_iterator_base &__x) const noexcept
- void _M_invalidate ()

- void _M_reset () noexcept
- const _Safe_unordered_container_base * **_M_safe_container** () const noexcept
- bool _M_singular () const noexcept
- void _M_unlink () noexcept

Public Attributes

- _Safe_iterator_base * _M_next
- _Safe_iterator_base * _M_prior
- const _Safe_sequence_base * _M_sequence
- unsigned int _M_version

Protected Member Functions

- _Safe_local_iterator_base ()
- _Safe_local_iterator_base (const _Safe_local_iterator_base &__x, bool __constant)
- _Safe_local_iterator_base (const _Safe_unordered_container_base *__seq, bool __constant)
- void _M_attach (const _Safe_sequence_base *__seq, bool __constant)
- void _M_attach (const _Safe_unordered_container_base *__cont, bool __constant)
- void _M_attach_single (const _Safe_sequence_base *__seq, bool __constant) noexcept
- void _M_attach_single (const _Safe_unordered_container_base *__cont, bool __constant) noexcept
- void _M_detach ()
- void _M_detach_single () noexcept
- __gnu_cxx::__mutex & _M_get_mutex () noexcept

6.153.1 Detailed Description

Basic functionality for a *safe* iterator.
The _Safe_local_iterator_base base class implements the functionality of a safe local iterator that is not specific to a particular iterator type. It contains a pointer back to the container it references along with iterator version information and pointers to form a doubly-linked list of local iterators referenced by the container.
This class must not perform any operations that can throw an exception, or the exception guarantees of derived iterators will be broken.

6.153.2 Constructor & Destructor Documentation

_Safe_local_iterator_base() [1/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base () [inline], [protected]
```

Initializes the iterator and makes it singular.
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), and _Safe_local_iterator_base().

_Safe_local_iterator_base() [2/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (
            const _Safe_unordered_container_base * __seq,
            bool __constant) [inline], [protected]
```

Initialize the iterator to reference the container pointed to by __seq. __constant is true when we are initializing a constant local iterator, and false if it is a mutable local iterator. Note that __seq may be NULL, in which case the iterator will be singular. Otherwise, the iterator will reference __seq and be nonsingular.
References _M_attach().

_Safe_local_iterator_base() [3/3]

```
__gnu_debug::_Safe_local_iterator_base::_Safe_local_iterator_base (
            const _Safe_local_iterator_base & __x,
            bool __constant)  [inline], [protected]
```

Initializes the iterator to reference the same container that __x does. __constant is true if this is a constant iterator, and false if it is mutable.
References _Safe_local_iterator_base(), and _M_attach().

6.153.3 Member Function Documentation

_M_attach() [1/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [inherited]
```

Attaches this iterator to the given sequence, detaching it from whatever sequence it was attached to originally. If the new sequence is the NULL pointer, the iterator is left unattached.
References _M_attach().
Referenced by _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach(), and _M_attach().

_M_attach() [2/2]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach (
            const _Safe_unordered_container_base * __cont,
            bool __constant)  [protected]
```

Attaches this iterator to the given container, detaching it from whatever container it was attached to originally. If the new container is the NULL pointer, the iterator is left unattached.
Referenced by _Safe_local_iterator_base(), _Safe_local_iterator_base(), and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UCon

_M_attach_single() [1/2]

```
void __gnu_debug::_Safe_iterator_base::_M_attach_single (
            const _Safe_sequence_base * __seq,
            bool __constant)  [protected], [noexcept], [inherited]
```

Likewise, but not thread-safe.
References _M_attach_single().
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_attach_single(), and _M_attach_single().

_M_attach_single() [2/2]

```
void __gnu_debug::_Safe_local_iterator_base::_M_attach_single (
            const _Safe_unordered_container_base * __cont,
            bool __constant)  [protected], [noexcept]
```

Likewise, but not thread-safe.
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_attach_single().

_M_attached_to()

```
bool __gnu_debug::_Safe_iterator_base::_M_attached_to (
            const _Safe_sequence_base * __seq) const  [inline], [inherited]
```

Determines if we are attached to the given sequence.
References _M_attached_to(), and _M_sequence.
Referenced by _M_attached_to().

_M_can_compare()

```
bool __gnu_debug::_Safe_iterator_base::_M_can_compare (
            const _Safe_iterator_base & __x) const  [noexcept], [inherited]
```

Can we compare this iterator to the given iterator __x? Returns true if both iterators are nonsingular and reference the same sequence.
References _Safe_iterator_base(), and _M_can_compare().
Referenced by _M_can_compare().

_M_detach()

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach ()  [protected]
```

Detach the iterator for whatever container it is attached to, if any.
Referenced by __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_ite
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_detach_single()

```
void __gnu_debug::_Safe_local_iterator_base::_M_detach_single ()  [protected], [noexcept]
```

Likewise, but not thread-safe.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_iterator_base::_M_get_mutex ()  [protected], [noexcept],
[inherited]
```

For use in _Safe_iterator.
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator++(), __gnu_debug::_Safe_local_iterator< _OtherIterator,
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_invalidate()

```
void __gnu_debug::_Safe_iterator_base::_M_invalidate ()  [inline], [inherited]
```

Invalidate the iterator, making it singular.
References _M_invalidate(), and _M_version.
Referenced by _M_invalidate().

_M_reset()

```
void __gnu_debug::_Safe_iterator_base::_M_reset ()  [noexcept], [inherited]
```

Reset all member variables

_M_singular()

```
bool __gnu_debug::_Safe_iterator_base::_M_singular () const  [noexcept], [inherited]
```

Is this iterator singular?
Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m
__gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContaine
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_Safe_local_iterator(), __gnu_debug::_Safe_local_iterator< _Other
__gnu_debug::__check_singular_aux(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::_M_dereferenceable(),
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_dereferenceable(), __gnu_debug::_Safe_iterator< _Base_itera
__gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::_M_incrementable(), __gnu_debug::_Safe_iterator< _Base_iterator
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

_M_unlink()

```
void __gnu_debug::_Safe_iterator_base::_M_unlink () [inline], [noexcept], [inherited]
```

Unlink itself

References _M_next, _M_prior, and _M_unlink().

Referenced by _M_unlink().

6.153.4 Member Data Documentation

_M_next

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_next [inherited]
```

Pointer to the next iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _Sequence >::_M and _M_unlink().

_M_prior

```
_Safe_iterator_base* __gnu_debug::_Safe_iterator_base::_M_prior [inherited]
```

Pointer to the previous iterator in the sequence's list of iterators. Only valid when _M_sequence != NULL.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), __gnu_debug::_Safe_sequence< _Sequence >::_M and _M_unlink().

_M_sequence

```
const _Safe_sequence_base* __gnu_debug::_Safe_iterator_base::_M_sequence [inherited]
```

The sequence this iterator references; may be NULL to indicate a singular iterator. Stored as pointer-to-const because sequence could be declared as const.

Referenced by __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), __gnu_debug::_Safe_iterator< _Base_iterator, m __gnu_debug::_Safe_iterator< _Base_iterator, map >::_Safe_iterator(), _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_attached_to(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=(), and __gnu_debug::_Safe_local_iterator< _OtherIterato

_M_version

```
unsigned int __gnu_debug::_Safe_iterator_base::_M_version [inherited]
```

The version number of this iterator. The sentinel value 0 is used to indicate an invalidated iterator (i.e., one that is singular because of an operation on the container). This version number must equal the version number in the sequence referenced by _M_sequence for the iterator to be non-singular.

Referenced by _Safe_iterator_base(), _Safe_iterator_base(), _Safe_iterator_base(), _M_invalidate(), and __gnu_debug::_Safe_sequence

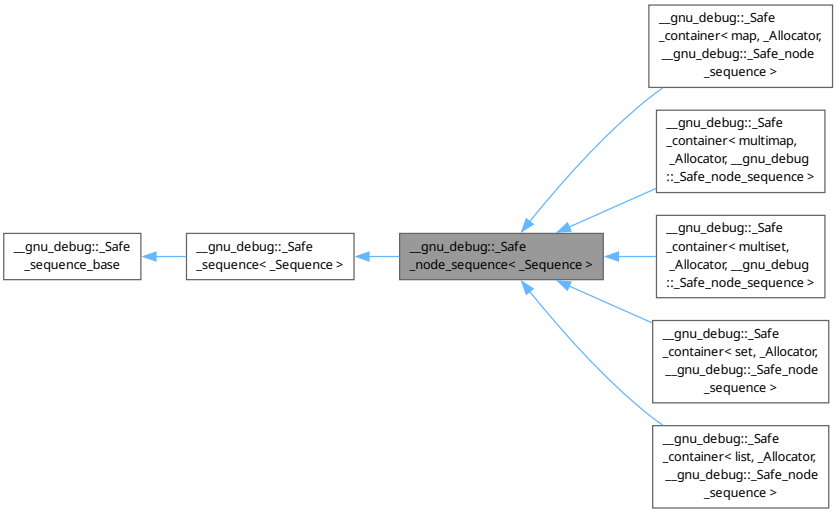
The documentation for this class was generated from the following file:

- safe_unordered_base.h

6.154 __gnu_debug::_Safe_node_sequence< _Sequence > Class Template Reference

```
#include <safe_sequence.h>
```

Inheritance diagram for __gnu_debug::_Safe_node_sequence< _Sequence >:



Public Member Functions

- **_Safe_node_sequence** (_Safe_node_sequence &&)=default
- **_Safe_node_sequence** (_Safe_node_sequence const &)=default
- template<typename _Predicate>
  constexpr void _M_invalidate_if (_Predicate __pred) const
- template<typename _Predicate>
  void _M_transfer_from_if (const _Safe_sequence &__from, _Predicate __pred) const
- constexpr _Safe_node_sequence & **operator=** (_Safe_node_sequence &&__x) noexcept
- constexpr _Safe_node_sequence & **operator=** (const _Safe_node_sequence &) noexcept

Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all () const
- void _M_detach_singular () const
- __gnu_cxx::__mutex & _M_get_mutex () const noexcept
- constexpr void **_M_invalidate_all** () const
- void _M_revalidate_singular () const
- void _M_swap (const _Safe_sequence_base &__x) const noexcept

6.154.1 Detailed Description

template<typename _Sequence>
class __gnu_debug::_Safe_node_sequence< _Sequence >

Like _Safe_sequence but with a special _M_invalidate_all implementation not invalidating past-the-end iterators. Used by node based sequence.

6.154.2 Member Function Documentation

_M_detach_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () const  [protected], [inherited]
```

Detach all iterators, leaving them singular.
Referenced by ~_Safe_sequence_base().

_M_detach_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const  [protected], [inherited]
```

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const  [protected], [noexcept],
[inherited]
```

For use in _Safe_sequence.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_invalidate_if()

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if (
            _Predicate __pred) const  [inline], [constexpr], [inherited]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

_M_revalidate_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const  [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

_M_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
            const _Safe_sequence_base & __x) const  [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

_M_transfer_from_if()

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if (
            const _Safe_sequence< _Sequence > & __from,
            _Predicate __pred) const  [inherited]
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.
References std::__addressof(), __gnu_debug::_Safe_sequence_base::_M_const_iterators, __gnu_debug::_Safe_sequence_base::_M_ge __gnu_debug::_Safe_sequence_base::_M_iterators, __gnu_debug::_Safe_iterator_base::_M_next, __gnu_debug::_Safe_iterator_base::_ __gnu_debug::_Safe_iterator_base::_M_sequence, __gnu_debug::_Safe_iterator_base::_M_version, and __gnu_debug::_Safe_sequenc

6.154.3 Member Data Documentation

_M_const_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators  [mutable], [inherited]
```

The list of constant iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators  [mutable], [inherited]
```

The list of mutable iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version  [mutable], [inherited]
```

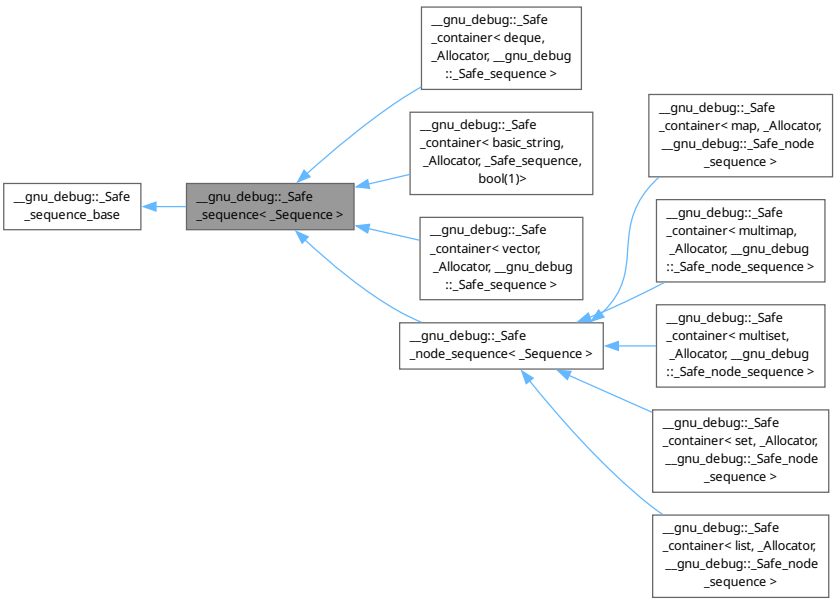
The container version number. This number may never be 0.
Referenced by _M_invalidate_all(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer > and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().
The documentation for this class was generated from the following file:

- safe_sequence.h

6.155 __gnu_debug::_Safe_sequence< _Sequence > Class Template Reference

```
#include <safe_sequence.h>
```

Inheritance diagram for __gnu_debug::_Safe_sequence< _Sequence >:



Public Member Functions

- template<typename _Predicate>
  constexpr void _M_invalidate_if (_Predicate __pred) const
- template<typename _Predicate>
  void _M_transfer_from_if (const _Safe_sequence &__from, _Predicate __pred) const

Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all () const
- void _M_detach_singular () const
- __gnu_cxx::__mutex & _M_get_mutex () const noexcept
- constexpr void _M_invalidate_all () const
- void _M_revalidate_singular () const
- void _M_swap (const _Safe_sequence_base &__x) const noexcept

6.155.1 Detailed Description

template<typename _Sequence>
class __gnu_debug::_Safe_sequence< _Sequence >

Base class for constructing a *safe* sequence type that tracks iterators that reference it.

The class template _Safe_sequence simplifies the construction of *safe* sequences that track the iterators that reference the sequence, so that the iterators are notified of changes in the sequence that may affect their operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template _Safe_iterator for this sequence. Iterators will then be tracked automatically.

6.155.2 Member Function Documentation

_M_detach_all()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_all () const  [protected], [inherited]
```

Detach all iterators, leaving them singular.
Referenced by ~_Safe_sequence_base().

_M_detach_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const  [protected], [inherited]
```

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const  [protected], [noexcept],
[inherited]
```

For use in _Safe_sequence.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_invalidate_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const  [inline], [constexpr], [protected],
[inherited]
```

Invalidates all iterators.
References _M_invalidate_all(), and _M_version.
Referenced by _M_invalidate_all().

_M_invalidate_if()

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence< _Sequence >::_M_invalidate_if (
            _Predicate __pred) const  [inline], [constexpr]
```

Invalidates all iterators `x` that reference this sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

_M_revalidate_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const  [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

_M_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
            const _Safe_sequence_base & __x) const  [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

_M_transfer_from_if()

```
template<typename _Sequence>
template<typename _Predicate>
void __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if (
            const _Safe_sequence< _Sequence > & __from,
            _Predicate __pred) const
```

Transfers all iterators `x` that reference `from` sequence, are not singular, and for which `__pred(x)` returns `true`. `__pred` will be invoked with the normal iterators nested in the safe ones.

References std::__addressof(), __gnu_debug::_Safe_sequence_base::_M_const_iterators, __gnu_debug::_Safe_sequence_base::_M_ge __gnu_debug::_Safe_sequence_base::_M_iterators, __gnu_debug::_Safe_iterator_base::_M_next, __gnu_debug::_Safe_iterator_base::_ __gnu_debug::_Safe_iterator_base::_M_sequence, __gnu_debug::_Safe_iterator_base::_M_version, and __gnu_debug::_Safe_sequence

6.155.3 Member Data Documentation

_M_const_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators  [mutable], [inherited]
```

The list of constant iterators that reference this container.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators  [mutable], [inherited]
```

The list of mutable iterators that reference this container.

Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version  [mutable], [inherited]
```

The container version number. This number may never be 0.

Referenced by _M_invalidate_all(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator __gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer > and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().

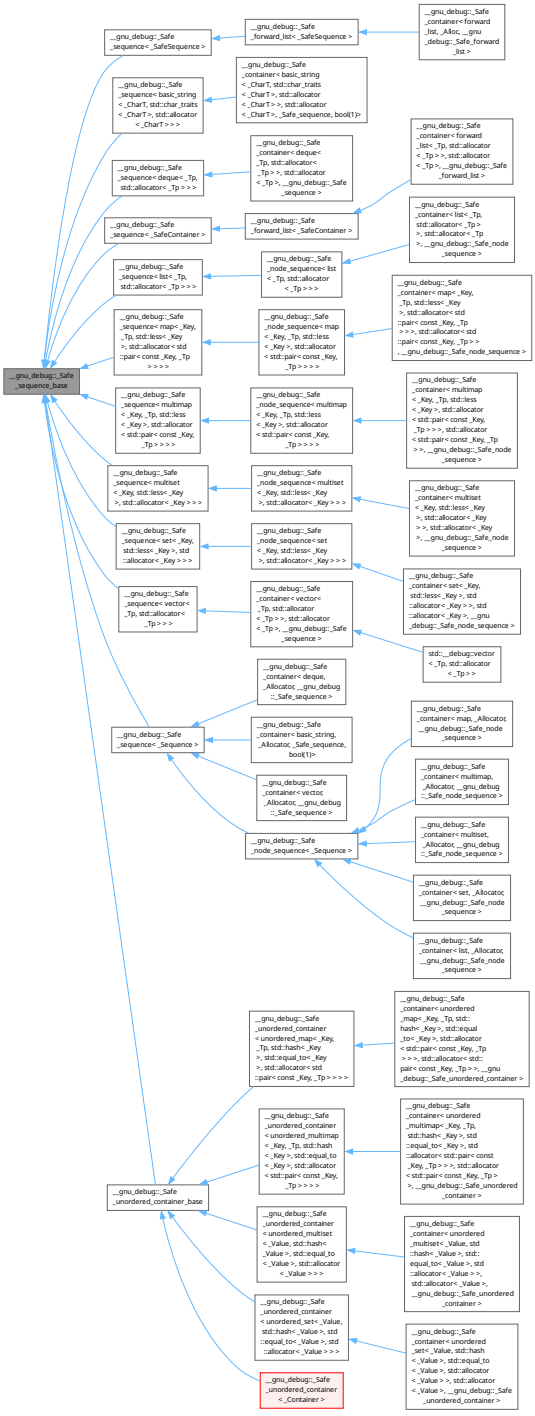
The documentation for this class was generated from the following files:

- formatter.h
- safe_sequence.h
- safe_sequence.tcc

6.156 __gnu_debug::_Safe_sequence_base Class Reference

```
#include <safe_base.h>
```

Inheritance diagram for __gnu_debug::_Safe_sequence_base:



Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_iterators
- unsigned int _M_version

Protected Member Functions

- constexpr **_Safe_sequence_base** (_Safe_sequence_base &&__seq) noexcept
- constexpr **_Safe_sequence_base** (const _Safe_sequence_base &) noexcept
- constexpr ~_Safe_sequence_base () noexcept
- void _M_detach_all () const
- void _M_detach_singular () const
- __gnu_cxx::__mutex & _M_get_mutex () const noexcept
- constexpr void _M_invalidate_all () const
- void _M_revalidate_singular () const
- void _M_swap (const _Safe_sequence_base &__x) const noexcept
- constexpr _Safe_sequence_base & **operator=** (_Safe_sequence_base &&__x) noexcept
- constexpr _Safe_sequence_base & **operator=** (const _Safe_sequence_base &) noexcept

Friends

- class **_Safe_iterator_base**

6.156.1 Detailed Description

Base class that supports tracking of iterators that reference a sequence.
The _Safe_sequence_base class provides basic support for tracking iterators into a sequence. Sequences that track iterators must derived from _Safe_sequence_base publicly, so that safe iterators (which inherit _Safe_iterator_base) can attach to them. This class contains two linked lists of iterators, one for constant iterators and one for mutable iterators, and a version number that allows very fast invalidation of all iterators that reference the container.
This class must ensure that no operation on it may throw an exception, otherwise *safe* sequences may fail to provide the exception-safety guarantees required by the C++ standard.

6.156.2 Constructor & Destructor Documentation

~_Safe_sequence_base()

`__gnu_debug::_Safe_sequence_base::~_Safe_sequence_base () [inline], [constexpr], [protected], [noexcept]`
Notify all iterators that reference this sequence that the sequence is being destroyed.
References _M_detach_all().

6.156.3 Member Function Documentation

_M_detach_all()

`void __gnu_debug::_Safe_sequence_base::_M_detach_all () const [protected]`
Detach all iterators, leaving them singular.
Referenced by ~_Safe_sequence_base().

_M_detach_singular()

`void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const [protected]`
Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const  [protected], [noexcept]
```

For use in _Safe_sequence.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_invalidate_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const  [inline], [constexpr], [protected]
```

Invalidates all iterators.
References _M_invalidate_all(), and _M_version.
Referenced by _M_invalidate_all().

_M_revalidate_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const  [protected]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

_M_swap()

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
             const _Safe_sequence_base & __x) const  [protected], [noexcept]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

6.156.4 Member Data Documentation

_M_const_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators  [mutable]
```

The list of constant iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators  [mutable]
```

The list of mutable iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version  [mutable]
```

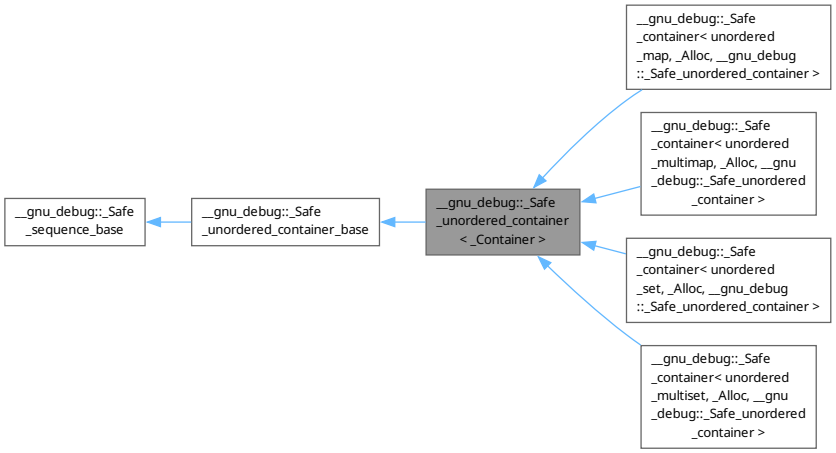
The container version number. This number may never be 0.
Referenced by _M_invalidate_all(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().
The documentation for this class was generated from the following file:

- safe_base.h

6.157 __gnu_debug::_Safe_unordered_container< _Container > Class Template Reference

```
#include <safe_unordered_container.h>
```

Inheritance diagram for __gnu_debug::_Safe_unordered_container< _Container >:



Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_const_local_iterators
- _Safe_iterator_base * _M_iterators
- _Safe_iterator_base * _M_local_iterators
- unsigned int _M_version

Protected Member Functions

- void _M_detach_all () const
- void _M_detach_singular () const
- __gnu_cxx::__mutex & _M_get_mutex () const noexcept
- template<typename _VictimIt>
  void **_M_invalidate** (_VictimIt __victim)
- template<typename _VictimIt>
  void **_M_invalidate** (_VictimIt __victim, const __gnu_cxx::__scoped_lock &__lock)
- constexpr void _M_invalidate_all () const
- void **_M_invalidate_all** ()
- template<typename _Predicate>
  void **_M_invalidate_all_if** (_Predicate __pred)
- void _M_revalidate_singular () const
- void _M_swap (const _Safe_sequence_base &__x) const noexcept
- void _M_swap (const _Safe_unordered_container_base &__x) const noexcept

6.157.1 Detailed Description

template<typename _Container>
class __gnu_debug::_Safe_unordered_container< _Container >

Base class for constructing a *safe* unordered container type that tracks iterators that reference it.
The class template _Safe_unordered_container simplifies the construction of *safe* unordered containers that track the iterators that reference the container, so that the iterators are notified of changes in the container that may affect their

operation, e.g., if the container invalidates its iterators or is destructed. This class template may only be used by deriving from it and passing the name of the derived class as its template parameter via the curiously recurring template pattern. The derived class must have `iterator` and `const_iterator` types that are instantiations of class template _↩ Safe_iterator for this container and `local_iterator` and `const_local_iterator` types that are instantiations of class template _Safe_local_iterator for this container. Iterators will then be tracked automatically.

6.157.2 Member Function Documentation

_M_detach_all()

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all () const  [protected], [inherited]
```

Detach all iterators, leaving them singular.
Referenced by ~_Safe_unordered_container_base().

_M_detach_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const  [protected], [inherited]
```

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const  [protected], [noexcept],
[inherited]
```

For use in _Safe_sequence.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_invalidate_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const  [inline], [constexpr], [protected],
[inherited]
```

Invalidates all iterators.
References _M_invalidate_all(), and _M_version.
Referenced by _M_invalidate_all().

_M_revalidate_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const  [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

_M_swap() [1/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
            const _Safe_sequence_base & __x) const  [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

_M_swap() [2/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
            const _Safe_unordered_container_base & __x) const  [protected], [noexcept], [inherited]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

6.157.3 Member Data Documentation

_M_const_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators  [mutable], [inherited]
```

The list of constant iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_const_local_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators  [mutable],
[inherited]
```

The list of constant local iterators that reference this container.

_M_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators  [mutable], [inherited]
```

The list of mutable iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_local_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators  [mutable],
[inherited]
```

The list of mutable local iterators that reference this container.

_M_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version  [mutable], [inherited]
```

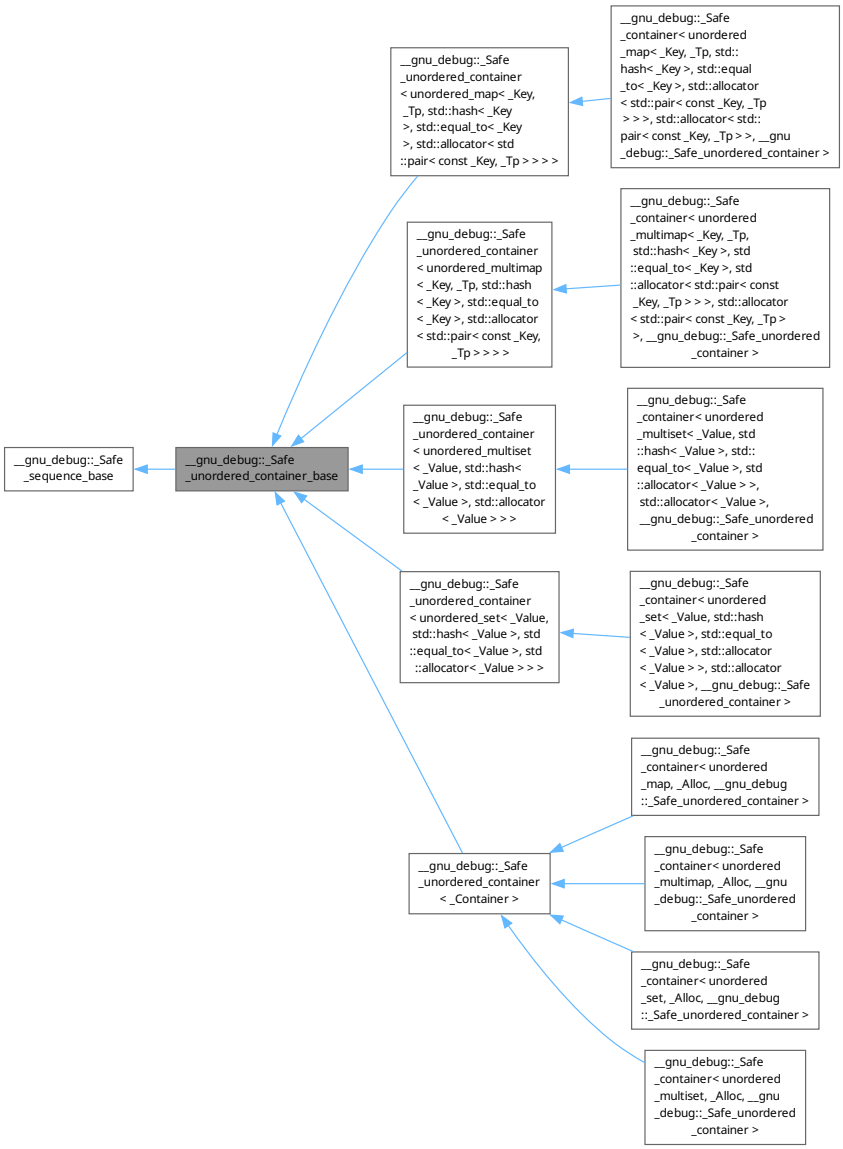
The container version number. This number may never be 0.
Referenced by _M_invalidate_all(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().
The documentation for this class was generated from the following files:

- safe_unordered_container.h
- safe_unordered_container.tcc

6.158 __gnu_debug::_Safe_unordered_container_base Class Reference

```
#include <safe_unordered_base.h>
```

Inheritance diagram for __gnu_debug::_Safe_unordered_container_base:



Public Attributes

- _Safe_iterator_base * _M_const_iterators
- _Safe_iterator_base * _M_const_local_iterators
- _Safe_iterator_base * _M_iterators
- _Safe_iterator_base * _M_local_iterators
- unsigned int _M_version

Protected Member Functions

- **_Safe_unordered_container_base** (_Safe_unordered_container_base &&__x) noexcept

- **_Safe_unordered_container_base** (const _Safe_unordered_container_base &) noexcept
- ~_Safe_unordered_container_base () noexcept
- void _M_detach_all () const
- void _M_detach_singular () const
- __gnu_cxx::__mutex & _M_get_mutex () const noexcept
- constexpr void _M_invalidate_all () const
- void _M_revalidate_singular () const
- void _M_swap (const _Safe_sequence_base &__x) const noexcept
- void _M_swap (const _Safe_unordered_container_base &__x) const noexcept
- _Safe_unordered_container_base & **operator=** (_Safe_unordered_container_base &&)=default
- _Safe_unordered_container_base & **operator=** (_Safe_unordered_container_base const &)=default

Friends

- class **_Safe_local_iterator_base**

6.158.1 Detailed Description

Base class that supports tracking of local iterators that reference an unordered container.
The _Safe_unordered_container_base class provides basic support for tracking iterators into an unordered container. Containers that track iterators must derived from _Safe_unordered_container_base publicly, so that safe iterators (which inherit _Safe_iterator_base) can attach to them. This class contains four linked lists of iterators, one for constant iterators, one for mutable iterators, one for constant local iterators, one for mutable local iterators and a version number that allows very fast invalidation of all iterators that reference the container.
This class must ensure that no operation on it may throw an exception, otherwise *safe* containers may fail to provide the exception-safety guarantees required by the C++ standard.

6.158.2 Constructor & Destructor Documentation

~_Safe_unordered_container_base()

```
__gnu_debug::_Safe_unordered_container_base::~_Safe_unordered_container_base ()  [inline], [protected],
[noexcept]
```

Notify all iterators that reference this container that the container is being destroyed.
References _M_detach_all().

6.158.3 Member Function Documentation

_M_detach_all()

```
void __gnu_debug::_Safe_unordered_container_base::_M_detach_all () const  [protected]
```

Detach all iterators, leaving them singular.
Referenced by ~_Safe_unordered_container_base().

_M_detach_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_detach_singular () const  [protected], [inherited]
```

Detach all singular iterators.

Postcondition

for all iterators i attached to this sequence, i->_M_version == _M_version.

_M_get_mutex()

```
__gnu_cxx::__mutex & __gnu_debug::_Safe_sequence_base::_M_get_mutex () const  [protected], [noexcept], [inherited]
```

For use in _Safe_sequence.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_invalidate_all()

```
void __gnu_debug::_Safe_sequence_base::_M_invalidate_all () const  [inline], [constexpr], [protected], [inherited]
```

Invalidates all iterators.
References _M_invalidate_all(), and _M_version.
Referenced by _M_invalidate_all().

_M_revalidate_singular()

```
void __gnu_debug::_Safe_sequence_base::_M_revalidate_singular () const  [protected], [inherited]
```

Revalidates all attached singular iterators. This method may be used to validate iterators that were invalidated before (but for some reason, such as an exception, need to become valid again).

_M_swap() [1/2]

```
void __gnu_debug::_Safe_sequence_base::_M_swap (
            const _Safe_sequence_base & __x) const  [protected], [noexcept], [inherited]
```

Swap this sequence with the given sequence. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

_M_swap() [2/2]

```
void __gnu_debug::_Safe_unordered_container_base::_M_swap (
            const _Safe_unordered_container_base & __x) const  [protected], [noexcept]
```

Swap this container with the given container. This operation also swaps ownership of the iterators, so that when the operation is complete all iterators that originally referenced one container now reference the other container.

6.158.4 Member Data Documentation

_M_const_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_const_iterators  [mutable], [inherited]
```

The list of constant iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_const_local_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_const_local_iterators  [mutable]
```

The list of constant local iterators that reference this container.

_M_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_sequence_base::_M_iterators  [mutable], [inherited]
```

The list of mutable iterators that reference this container.
Referenced by __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if().

_M_local_iterators

```
_Safe_iterator_base* __gnu_debug::_Safe_unordered_container_base::_M_local_iterators  [mutable]
```
The list of mutable local iterators that reference this container.

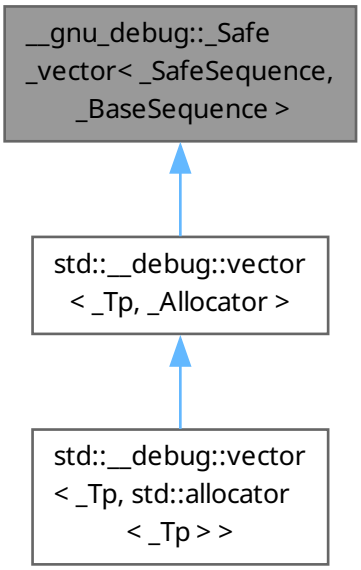
_M_version

```
unsigned int __gnu_debug::_Safe_sequence_base::_M_version  [mutable], [inherited]
```
The container version number. This number may never be 0.
Referenced by _M_invalidate_all(), __gnu_debug::_Safe_sequence< _Sequence >::_M_transfer_from_if(), __gnu_debug::_Safe_iterator
__gnu_debug::_Safe_iterator< _Base_iterator, map >::operator=(), __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >
and __gnu_debug::_Safe_local_iterator< _OtherIterator, _UContainer >::operator=().
The documentation for this class was generated from the following file:

- safe_unordered_base.h

6.159 __gnu_debug::_Safe_vector< _SafeSequence, _BaseSequence > Class Template Reference

```
#include <vector>
```
Inheritance diagram for __gnu_debug::_Safe_vector< _SafeSequence, _BaseSequence >:



Protected Member Functions

- constexpr **_Safe_vector** (_Safe_vector &&__x) noexcept
- constexpr **_Safe_vector** (const _Safe_vector &) noexcept
- constexpr **_Safe_vector** (size_type __n) noexcept
- bool **_M_requires_reallocation** (size_type __elements) const noexcept
- constexpr void **_M_update_guaranteed_capacity** () noexcept

- constexpr _Safe_vector & **operator=** (_Safe_vector &&__x) noexcept
- constexpr _Safe_vector & **operator=** (const _Safe_vector &) noexcept

Protected Attributes

- size_type **_M_guaranteed_capacity**

6.159.1 Detailed Description

template<typename _SafeSequence, typename _BaseSequence>
class __gnu_debug::_Safe_vector< _SafeSequence, _BaseSequence >

Base class for Debug Mode vector.
Adds information about the guaranteed capacity, which is useful for detecting code which relies on non-portable implementation details of the libstdc++ reallocation policy.
The documentation for this class was generated from the following file:

- debug/vector

6.160 __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- void **operator()** (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)

6.160.1 Detailed Description

template<bool __stable, class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >

Stable sorting functor.
Used to reduce code instantiation in multiway_merge_sampling_splitting.
The documentation for this struct was generated from the following file:

- multiway_merge.h

6.161 __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering > Struct Template Reference

`#include <multiway_merge.h>`

Public Member Functions

- void **operator()** (_RAIter __first, _RAIter __last, _StrictWeakOrdering __comp)

6.161.1 Detailed Description

template<class _RAIter, class _StrictWeakOrdering>
struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >

Non-__stable sorting functor.
Used to reduce code instantiation in multiway_merge_sampling_splitting.
The documentation for this struct was generated from the following file:

- multiway_merge.h

6.162 std::__detail::_Scanner< _CharT > Class Template Reference

`#include <regex_scanner.h>`

Public Types

- typedef const std::ctype< _CharT > **_CtypeT**
- typedef regex_constants::syntax_option_type **_FlagT**
- typedef std::basic_string< _CharT > **_StringT**
- enum **_TokenT** : unsigned {
  _S_token_anychar , **_S_token_ord_char** , **_S_token_oct_num** , **_S_token_hex_num** ,
  _S_token_backref , **_S_token_subexpr_begin** , **_S_token_subexpr_no_group_begin** , **_S_token_subexpr_lookahead_begin** ,
  _S_token_subexpr_end , **_S_token_bracket_begin** , **_S_token_bracket_neg_begin** , **_S_token_bracket_end** ,
  _S_token_interval_begin , **_S_token_interval_end** , **_S_token_quoted_class** , **_S_token_char_class_name** ,
  _S_token_collsymbol , **_S_token_equiv_class_name** , **_S_token_opt** , **_S_token_or** ,
  _S_token_closure0 , **_S_token_closure1** , **_S_token_line_begin** , **_S_token_line_end** ,
  _S_token_word_bound , **_S_token_comma** , **_S_token_dup_count** , **_S_token_eof** ,
  _S_token_bracket_dash , **_S_token_unknown** }

Public Member Functions

- **_Scanner** (const _CharT *__begin, const _CharT *__end, _FlagT __flags, std::locale __loc)
- void **_M_advance** ()
- _TokenT **_M_get_token** () const noexcept
- const _StringT & **_M_get_value** () const noexcept

Protected Types

- enum **_StateT** { **_S_state_normal** , **_S_state_in_brace** , **_S_state_in_bracket** }

Protected Member Functions

- const char * **_M_find_escape** (char __c)
- bool **_M_is_awk** () const
- bool **_M_is_basic** () const
- bool **_M_is_ecma** () const
- bool **_M_is_extended** () const
- bool **_M_is_grep** () const

Protected Attributes

- bool **_M_at_bracket_start**
- const std::pair< char, char > **_M_awk_escape_tbl** [11]
- const char * **_M_basic_spec_char**
- const std::pair< char, char > **_M_ecma_escape_tbl** [8]
- const char * **_M_ecma_spec_char**
- const std::pair< char, char > * **_M_escape_tbl**
- const char * **_M_extended_spec_char**
- _FlagT **_M_flags**
- const char * **_M_spec_char**
- _StateT **_M_state**
- _TokenT **_M_token**
- const std::pair< char, _TokenT > **_M_token_tbl** [9]

6.162.1 Detailed Description

template<typename _CharT>
class std::__detail::_Scanner< _CharT >

Scans an input range for regex tokens.
The _Scanner class interprets the regular expression pattern in the input range passed to its constructor as a sequence of parse tokens passed to the regular expression compiler. The sequence of tokens provided depends on the flag settings passed to the constructor: different regular expression grammars will interpret the same input pattern in syntactically different ways.

6.162.2 Member Enumeration Documentation

_TokenT

```
enum std::__detail::_ScannerBase::_TokenT : unsigned  [inherited]
```

Token types returned from the scanner.
The documentation for this class was generated from the following files:

- regex_scanner.h
- regex_scanner.tcc

6.163 __gnu_debug::_Sequence_traits< _Sequence > Struct Template Reference

```
#include <safe_iterator.h>
```

Public Types

- typedef _Distance_traits< typename _Sequence::iterator > **_DistTraits**

Static Public Member Functions

- static _DistTraits::__type **_S_size** (const _Sequence &__seq)

6.163.1 Detailed Description

template<typename _Sequence>
struct __gnu_debug::_Sequence_traits< _Sequence >

Sequence traits giving the size of a container if possible.
The documentation for this struct was generated from the following file:

- safe_iterator.h

6.164 __gnu_parallel::_Settings Struct Reference

```
#include <settings.h>
```

Static Public Member Functions

- static const _Settings & get () throw ()
- static void set (_Settings &) throw ()

Public Attributes

- _SequenceIndex accumulate_minimal_n
- unsigned int adjacent_difference_minimal_n
- _AlgorithmStrategy **algorithm_strategy**
- unsigned int cache_line_size
- _SequenceIndex count_minimal_n
- _SequenceIndex fill_minimal_n
- _FindAlgorithm **find_algorithm**
- double find_increasing_factor
- _SequenceIndex find_initial_block_size
- _SequenceIndex find_maximum_block_size
- float find_scale_factor
- _SequenceIndex find_sequential_search_size
- _SequenceIndex for_each_minimal_n
- _SequenceIndex generate_minimal_n
- unsigned long long L1_cache_size
- unsigned long long L2_cache_size
- _SequenceIndex max_element_minimal_n
- _SequenceIndex merge_minimal_n
- unsigned int merge_oversampling
- _SplittingAlgorithm **merge_splitting**
- _SequenceIndex min_element_minimal_n
- _MultiwayMergeAlgorithm **multiway_merge_algorithm**
- int multiway_merge_minimal_k
- _SequenceIndex multiway_merge_minimal_n
- unsigned int multiway_merge_oversampling
- _SplittingAlgorithm **multiway_merge_splitting**
- _SequenceIndex nth_element_minimal_n
- _SequenceIndex partial_sort_minimal_n
- _PartialSumAlgorithm **partial_sum_algorithm**
- float partial_sum_dilation
- unsigned int partial_sum_minimal_n
- double partition_chunk_share
- _SequenceIndex partition_chunk_size
- _SequenceIndex partition_minimal_n
- _SequenceIndex qsb_steals
- unsigned int random_shuffle_minimal_n
- _SequenceIndex replace_minimal_n
- _SequenceIndex search_minimal_n
- _SequenceIndex set_difference_minimal_n
- _SequenceIndex set_intersection_minimal_n
- _SequenceIndex set_symmetric_difference_minimal_n
- _SequenceIndex set_union_minimal_n
- _SortAlgorithm **sort_algorithm**
- _SequenceIndex sort_minimal_n
- unsigned int sort_mwms_oversampling
- unsigned int sort_qs_num_samples_preset
- _SequenceIndex sort_qsb_base_case_maximal_n
- _SplittingAlgorithm **sort_splitting**
- unsigned int TLB_size
- _SequenceIndex transform_minimal_n
- _SequenceIndex unique_copy_minimal_n
- _SequenceIndex **workstealing_chunk_size**

6.164.1 Detailed Description

class _Settings Run-time settings for the parallel mode including all tunable parameters.

6.164.2 Member Function Documentation

get()

```
const _Settings & __gnu_parallel::_Settings::get () throw ( )   [static]
```

Get the global settings.
Referenced by __gnu_parallel::__find_template(), __gnu_parallel::__find_template(), __gnu_parallel::__find_template(), __gnu_parallel::__for_each_template_random_access_workstealing(), __gnu_parallel::__parallel_nth_element(), __gnu_parallel::__parallel_partial_sum(), __gnu_parallel::__parallel_partial_sum_linear(), __gnu_parallel::__parallel_partition(), __gnu_parallel::__parallel_random_shuffle_drs(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort_qs_conquer(), __gnu_parallel::__qsb_local_sort_with_helping(), __gnu_parallel::__sequential_random_s __gnu_parallel::multiway_merge_sampling_splitting(), __gnu_parallel::parallel_multiway_merge(), __gnu_parallel::parallel_sort_mwms(), and __gnu_parallel::parallel_sort_mwms_pu().

set()

```
void __gnu_parallel::_Settings::set (
            _Settings & ) throw ( )   [static]
```

Set the global settings.
References accumulate_minimal_n, adjacent_difference_minimal_n, cache_line_size, count_minimal_n, fill_minimal_n, find_increasing_factor, find_initial_block_size, find_maximum_block_size, find_scale_factor, find_sequential_search_size, for_each_minimal_n, generate_minimal_n, L1_cache_size, L2_cache_size, max_element_minimal_n, merge_minimal_n, merge_oversampling, min_element_minimal_n, multiway_merge_minimal_k, multiway_merge_minimal_n, multiway_merge_oversampling nth_element_minimal_n, partial_sort_minimal_n, partial_sum_dilation, partial_sum_minimal_n, partition_chunk_share, partition_chunk_size, partition_minimal_n, qsb_steals, random_shuffle_minimal_n, replace_minimal_n, search_minimal_n, set(), set_difference_minimal_n, set_intersection_minimal_n, set_symmetric_difference_minimal_n, set_union_minimal_n, sort_minimal_n, sort_mwms_oversampling, sort_qs_num_samples_preset, sort_qsb_base_case_maximal_n, TLB_size, transform_minimal_n, and unique_copy_minimal_n.
Referenced by set().

6.164.3 Member Data Documentation

accumulate_minimal_n

```
_SequenceIndex __gnu_parallel::_Settings::accumulate_minimal_n
```

Minimal input size for accumulate.
Referenced by set().

adjacent_difference_minimal_n

```
unsigned int __gnu_parallel::_Settings::adjacent_difference_minimal_n
```

Minimal input size for adjacent_difference.
Referenced by set().

cache_line_size

```
unsigned int __gnu_parallel::_Settings::cache_line_size
```

Overestimation of cache line size. Used to avoid false sharing, i.e. elements of different threads are at least this amount apart.
Referenced by __gnu_parallel::__for_each_template_random_access_workstealing(), and set().

count_minimal_n

_SequenceIndex __gnu_parallel::_Settings::count_minimal_n
Minimal input size for count and count_if.
Referenced by set().

fill_minimal_n

_SequenceIndex __gnu_parallel::_Settings::fill_minimal_n
Minimal input size for fill.
Referenced by set().

find_increasing_factor

double __gnu_parallel::_Settings::find_increasing_factor
Block size increase factor for find.
Referenced by set().

find_initial_block_size

_SequenceIndex __gnu_parallel::_Settings::find_initial_block_size
Initial block size for find.
Referenced by __gnu_parallel::__find_template(), and set().

find_maximum_block_size

_SequenceIndex __gnu_parallel::_Settings::find_maximum_block_size
Maximal block size for find.
Referenced by set().

find_scale_factor

float __gnu_parallel::_Settings::find_scale_factor
Block size scale-down factor with respect to current position.
Referenced by __gnu_parallel::__find_template(), and set().

find_sequential_search_size

_SequenceIndex __gnu_parallel::_Settings::find_sequential_search_size
Start with looking for this many elements sequentially, for find.
Referenced by __gnu_parallel::__find_template(), __gnu_parallel::__find_template(), and set().

for_each_minimal_n

_SequenceIndex __gnu_parallel::_Settings::for_each_minimal_n
Minimal input size for for_each.
Referenced by set().

generate_minimal_n

_SequenceIndex __gnu_parallel::_Settings::generate_minimal_n
Minimal input size for generate.
Referenced by set().

L1_cache_size

`unsigned long long __gnu_parallel::_Settings::L1_cache_size`
size of the L1 cache in bytes (underestimation).
Referenced by set().

L2_cache_size

`unsigned long long __gnu_parallel::_Settings::L2_cache_size`
size of the L2 cache in bytes (underestimation).
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), __gnu_parallel::__sequential_random_shuffle(), and set().

max_element_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::max_element_minimal_n`
Minimal input size for max_element.
Referenced by set().

merge_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::merge_minimal_n`
Minimal input size for merge.
Referenced by set().

merge_oversampling

`unsigned int __gnu_parallel::_Settings::merge_oversampling`
Oversampling factor for merge.
Referenced by __gnu_parallel::multiway_merge_sampling_splitting(), __gnu_parallel::parallel_multiway_merge(), and set().

min_element_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::min_element_minimal_n`
Minimal input size for min_element.
Referenced by set().

multiway_merge_minimal_k

`int __gnu_parallel::_Settings::multiway_merge_minimal_k`
Oversampling factor for multiway_merge.
Referenced by set().

multiway_merge_minimal_n

`_SequenceIndex __gnu_parallel::_Settings::multiway_merge_minimal_n`
Minimal input size for multiway_merge.
Referenced by set().

multiway_merge_oversampling

`unsigned int __gnu_parallel::_Settings::multiway_merge_oversampling`
Oversampling factor for multiway_merge.
Referenced by set().

nth_element_minimal_n

_SequenceIndex __gnu_parallel::_Settings::nth_element_minimal_n
Minimal input size for nth_element.
Referenced by __gnu_parallel::__parallel_nth_element(), and set().

partial_sort_minimal_n

_SequenceIndex __gnu_parallel::_Settings::partial_sort_minimal_n
Minimal input size for partial_sort.
Referenced by set().

partial_sum_dilation

float __gnu_parallel::_Settings::partial_sum_dilation
Ratio for partial_sum. Assume "sum and write result" to be this factor slower than just "sum".
Referenced by __gnu_parallel::__parallel_partial_sum_linear(), and set().

partial_sum_minimal_n

unsigned int __gnu_parallel::_Settings::partial_sum_minimal_n
Minimal input size for partial_sum.
Referenced by set().

partition_chunk_share

double __gnu_parallel::_Settings::partition_chunk_share
Chunk size for partition, relative to input size. If > 0.0, this value overrides partition_chunk_size.
Referenced by __gnu_parallel::__parallel_partition(), and set().

partition_chunk_size

_SequenceIndex __gnu_parallel::_Settings::partition_chunk_size
Chunk size for partition.
Referenced by __gnu_parallel::__parallel_partition(), and set().

partition_minimal_n

_SequenceIndex __gnu_parallel::_Settings::partition_minimal_n
Minimal input size for partition.
Referenced by __gnu_parallel::__parallel_nth_element(), and set().

qsb_steals

_SequenceIndex __gnu_parallel::_Settings::qsb_steals
The number of stolen ranges in load-balanced quicksort.
Referenced by set().

random_shuffle_minimal_n

unsigned int __gnu_parallel::_Settings::random_shuffle_minimal_n
Minimal input size for random_shuffle.
Referenced by set().

replace_minimal_n

_SequenceIndex __gnu_parallel::_Settings::replace_minimal_n
Minimal input size for replace and replace_if.
Referenced by set().

search_minimal_n

_SequenceIndex __gnu_parallel::_Settings::search_minimal_n
Minimal input size for search and search_n.
Referenced by set().

set_difference_minimal_n

_SequenceIndex __gnu_parallel::_Settings::set_difference_minimal_n
Minimal input size for set_difference.
Referenced by set().

set_intersection_minimal_n

_SequenceIndex __gnu_parallel::_Settings::set_intersection_minimal_n
Minimal input size for set_intersection.
Referenced by set().

set_symmetric_difference_minimal_n

_SequenceIndex __gnu_parallel::_Settings::set_symmetric_difference_minimal_n
Minimal input size for set_symmetric_difference.
Referenced by set().

set_union_minimal_n

_SequenceIndex __gnu_parallel::_Settings::set_union_minimal_n
Minimal input size for set_union.
Referenced by set().

sort_minimal_n

_SequenceIndex __gnu_parallel::_Settings::sort_minimal_n
Minimal input size for parallel sorting.
Referenced by set().

sort_mwms_oversampling

unsigned int __gnu_parallel::_Settings::sort_mwms_oversampling
Oversampling factor for parallel std::sort (MWMS).
Referenced by __gnu_parallel::parallel_sort_mwms(), __gnu_parallel::parallel_sort_mwms_pu(), and set().

sort_qs_num_samples_preset

unsigned int __gnu_parallel::_Settings::sort_qs_num_samples_preset
Such many samples to take to find a good pivot (quicksort).
Referenced by set().

sort_qsb_base_case_maximal_n

_SequenceIndex __gnu_parallel::_Settings::sort_qsb_base_case_maximal_n
Maximal subsequence __length to switch to unbalanced __base case. Applies to std::sort with dynamically load-balanced quicksort.
Referenced by __gnu_parallel::__qsb_local_sort_with_helping(), and set().

TLB_size

unsigned int __gnu_parallel::_Settings::TLB_size
size of the Translation Lookaside Buffer (underestimation).
Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), __gnu_parallel::__sequential_random_shuffle(), and set().

transform_minimal_n

_SequenceIndex __gnu_parallel::_Settings::transform_minimal_n
Minimal input size for parallel std::transform.
Referenced by set().

unique_copy_minimal_n

_SequenceIndex __gnu_parallel::_Settings::unique_copy_minimal_n
Minimal input size for unique_copy.
Referenced by set().
The documentation for this struct was generated from the following file:

- settings.h

6.165 std::_Sp_ebo_helper< _Tp, false > Struct Template Reference

#include <shared_ptr_base.h>

Public Attributes

- _Tp **_M_obj**

6.165.1 Detailed Description

template<typename _Tp>
struct std::_Sp_ebo_helper< _Tp, false >

Specialization not using [[no_unique_address]].
The documentation for this struct was generated from the following file:

- shared_ptr_base.h

6.166 std::_Sp_ebo_helper< _Tp, true > Struct Template Reference

#include <shared_ptr_base.h>

Public Attributes

- _Tp **_M_obj**

6.166.1 Detailed Description

template<typename _Tp>
struct std::_Sp_ebo_helper< _Tp, true >

Specialization using [[no_unique_address]].
The documentation for this struct was generated from the following file:

- shared_ptr_base.h

6.167 __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference

`#include <multiway_mergesort.h>`

6.167.1 Detailed Description

template<bool __exact, typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >

Split consistently.
The documentation for this struct was generated from the following file:

- multiway_mergesort.h

6.168 __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference

`#include <multiway_mergesort.h>`

Public Member Functions

- void **operator()** (const _ThreadIndex __iam, _PMWMSSortingData< _RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits< _RAIter >::difference_type __num_samples) const

6.168.1 Detailed Description

template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >

Split by sampling.
The documentation for this struct was generated from the following file:

- multiway_mergesort.h

6.169 __gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator > Struct Template Reference

`#include <multiway_mergesort.h>`

Public Member Functions

- void **operator()** (const _ThreadIndex __iam, _PMWMSSortingData< _RAIter > *__sd, _Compare &__comp, const typename std::iterator_traits< _RAIter >::difference_type __num_samples) const

6.169.1 Detailed Description

template<typename _RAIter, typename _Compare, typename _SortingPlacesIterator>
struct __gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >

Split by exact splitting.
The documentation for this struct was generated from the following file:

- multiway_mergesort.h

6.170 std::__detail::_StateSeq< _TraitsT > Class Template Reference

`#include <regex_automaton.h>`

Public Types

- typedef _NFA< _TraitsT > **_RegexT**

Public Member Functions

- **_StateSeq** (_RegexT &__nfa, _StateIdT __s)
- **_StateSeq** (_RegexT &__nfa, _StateIdT __s, _StateIdT __end)
- void **_M_append** (_StateIdT __id)
- void **_M_append** (const _StateSeq &__s)
- _StateSeq **_M_clone** ()

Public Attributes

- _StateIdT **_M_end**
- _RegexT & **_M_nfa**
- _StateIdT **_M_start**

6.170.1 Detailed Description

template<typename _TraitsT>
class std::__detail::_StateSeq< _TraitsT >

Describes a sequence of one or more _State, its current start and end(s). This structure contains fragments of an NFA during construction.
The documentation for this class was generated from the following files:

- regex_automaton.h
- regex_automaton.tcc

6.171 __gnu_cxx::_Std_pointer_impl< _Tp > Class Template Reference

`#include <pointer.h>`

Public Types

- typedef _Tp **element_type**

Public Member Functions

- _Tp * **get** () const
- bool **operator<** (const _Std_pointer_impl &__rarg) const
- bool **operator==** (const _Std_pointer_impl &__rarg) const
- void **set** (element_type *__arg)

6.171.1 Detailed Description

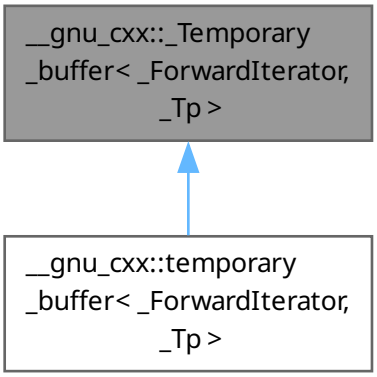
template<typename _Tp>
class __gnu_cxx::_Std_pointer_impl< _Tp >

A storage policy for use with _Pointer_adapter<> which yields a standard pointer.
A _Storage_policy is required to provide 4 things: 1) A get() API for returning the stored pointer value. 2) An set() API for storing a pointer value. 3) An element_type typedef to define the type this points to. 4) An operator<() to support pointer comparison. 5) An operator==() to support pointer comparison.
The documentation for this class was generated from the following file:

- pointer.h

6.172 __gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference

`#include <stl_tempbuf.h>`
Inheritance diagram for __gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >:



Public Types

- typedef pointer **iterator**
- typedef value_type * **pointer**
- typedef ptrdiff_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- _Temporary_buffer (_ForwardIterator __seed, size_type __original_len)
- size_type _M_requested_size () const
- iterator begin ()
- iterator end ()
- size_type size () const

Protected Attributes

- struct std::_Temporary_buffer::_Impl **_M_impl**
- size_type **_M_original_len**

6.172.1 Detailed Description

template<typename _ForwardIterator, typename _Tp>
class __gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >

This class is used in two places: stl_algo.h and ext/memory, where it is wrapped as the temporary_buffer class. See temporary_buffer docs for more notes.

6.172.2 Constructor & Destructor Documentation

_Temporary_buffer()

```
template<typename _ForwardIterator, typename _Tp>
std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer (
            _ForwardIterator __seed,
            size_type __original_len)
```

Constructs a temporary buffer of a size somewhere between zero and the given length.

6.172.3 Member Function Documentation

_M_requested_size()

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::_M_requested_size () const  [inline]
```

Returns the size requested by the constructor; may be >size().

begin()

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ()  [inline]
```

As per Table mumble.

end()

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end ()  [inline]
```

As per Table mumble.

size()

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const  [inline]
```

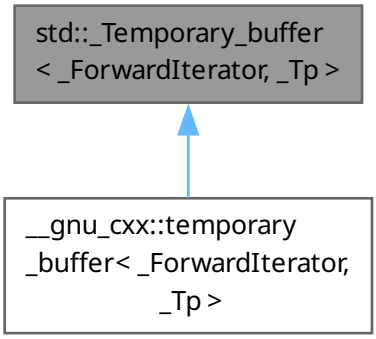
As per Table mumble.
The documentation for this class was generated from the following file:

- stl_tempbuf.h

6.173 std::_Temporary_buffer< _ForwardIterator, _Tp > Class Template Reference

```
#include <stl_tempbuf.h>
```

Inheritance diagram for std::_Temporary_buffer< _ForwardIterator, _Tp >:



Public Types

- typedef pointer **iterator**
- typedef value_type * **pointer**
- typedef ptrdiff_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- _Temporary_buffer (_ForwardIterator __seed, size_type __original_len)
- size_type _M_requested_size () const
- iterator begin ()
- iterator end ()
- size_type size () const

Protected Attributes

- struct std::_Temporary_buffer::_Impl **_M_impl**
- size_type **_M_original_len**

6.173.1 Detailed Description

template<typename _ForwardIterator, typename _Tp>
class std::_Temporary_buffer< _ForwardIterator, _Tp >

This class is used in two places: stl_algo.h and ext/memory, where it is wrapped as the temporary_buffer class. See temporary_buffer docs for more notes.

6.173.2 Constructor & Destructor Documentation

_Temporary_buffer()

```
template<typename _ForwardIterator, typename _Tp>
std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer (
```

```
_ForwardIterator __seed,
size_type __original_len)
```

Constructs a temporary buffer of a size somewhere between zero and the given length.
References begin(), and end().
Referenced by __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::temporary_buffer().

6.173.3 Member Function Documentation

_M_requested_size()

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::_M_requested_size () const  [inline]
```

Returns the size requested by the constructor; may be >size().

begin()

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ()  [inline]
```

As per Table mumble.
Referenced by _Temporary_buffer().

end()

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end ()  [inline]
```

As per Table mumble.
Referenced by _Temporary_buffer().

size()

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const  [inline]
```

As per Table mumble.
The documentation for this class was generated from the following file:

- stl_tempbuf.h

6.174 __gnu_cxx::_Unqualified_type< _Tp > Struct Template Reference

```
#include <pointer.h>
```

Public Types

- typedef _Tp **type**

6.174.1 Detailed Description

template<typename _Tp>
struct __gnu_cxx::_Unqualified_type< _Tp >

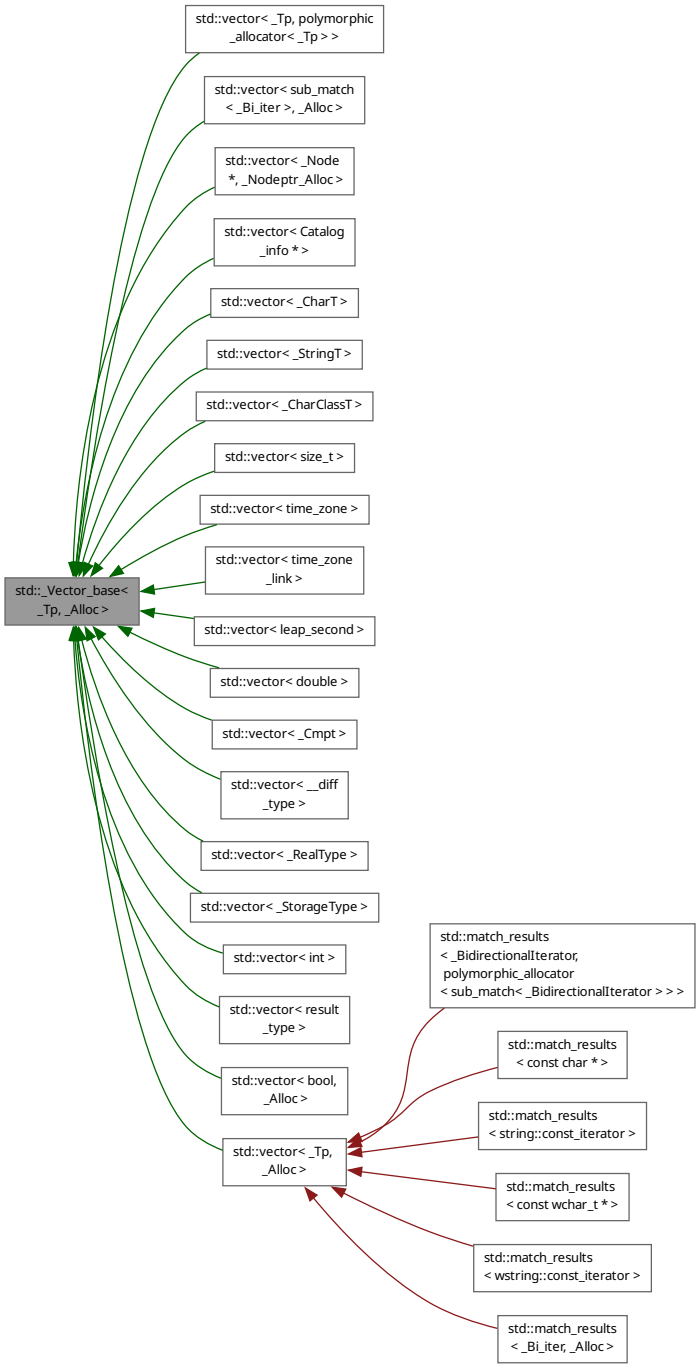
This structure accommodates the way in which std::iterator_traits<> is normally specialized for const T*, so that value_type is still T.
The documentation for this struct was generated from the following file:

- pointer.h

6.175 std::_Vector_base< _Tp, _Alloc > Struct Template Reference

`#include <stl_vector.h>`

Inheritance diagram for std::_Vector_base< _Tp, _Alloc >:



Public Types

- typedef __gnu_cxx::__alloc_traits< _Alloc >::template rebind< _Tp >::other **_Tp_alloc_type**
- typedef _Alloc **allocator_type**
- typedef __gnu_cxx::__alloc_traits< _Tp_alloc_type >::pointer **pointer**

Public Member Functions

- constexpr **_Vector_base** (_Tp_alloc_type &&__a) noexcept
- **_Vector_base** (_Vector_base &&)=default
- constexpr **_Vector_base** (_Vector_base &&__x, const allocator_type &__a)
- constexpr **_Vector_base** (const allocator_type &__a) noexcept
- constexpr **_Vector_base** (const allocator_type &__a, _Vector_base &&__x)
- constexpr **_Vector_base** (size_t __n)
- constexpr **_Vector_base** (size_t __n, const allocator_type &__a)
- constexpr pointer **_M_allocate** (size_t __n)
- constexpr _Alloc_result **_M_allocate_at_least** (size_t __n)
- constexpr void **_M_deallocate** (pointer __p, size_t __n)
- constexpr const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- constexpr _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- constexpr allocator_type **get_allocator** () const noexcept

Static Public Member Functions

- static constexpr size_t **_S_max_size** (const _Tp_alloc_type &__a) noexcept

Public Attributes

- _Vector_impl **_M_impl**

Protected Member Functions

- constexpr void **_M_create_storage** (size_t __n)

6.175.1 Detailed Description

template<typename _Tp, typename _Alloc>
struct std::_Vector_base< _Tp, _Alloc >

See bits/stl_deque.h's _Deque_base for an explanation.
The documentation for this struct was generated from the following file:

- stl_vector.h

6.176 std::add_const< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.176.1 Detailed Description

template<typename _Tp>
struct std::add_const< _Tp >

add_const
The documentation for this struct was generated from the following file:

- type_traits

6.177 std::add_cv< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.177.1 Detailed Description

template<typename _Tp>
struct std::add_cv< _Tp >

add_cv
The documentation for this struct was generated from the following file:

- type_traits

6.178 std::add_lvalue_reference< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.178.1 Detailed Description

template<typename _Tp>
struct std::add_lvalue_reference< _Tp >

add_lvalue_reference
The documentation for this struct was generated from the following file:

- type_traits

6.179 std::add_pointer< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.179.1 Detailed Description

template<**typename _Tp**>
struct std::add_pointer< **_Tp** >

add_pointer
The documentation for this struct was generated from the following file:

- type_traits

6.180 std::add_rvalue_reference< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.180.1 Detailed Description

template<**typename _Tp**>
struct std::add_rvalue_reference< **_Tp** >

add_rvalue_reference
The documentation for this struct was generated from the following file:

- type_traits

6.181 std::add_volatile< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.181.1 Detailed Description

template<**typename _Tp**>
struct std::add_volatile< **_Tp** >

add_volatile
The documentation for this struct was generated from the following file:

- type_traits

6.182 std::adopt_lock_t Struct Reference

`#include <std_mutex.h>`

6.182.1 Detailed Description

Assume the calling thread has already obtained mutex ownership and manage it.
The documentation for this struct was generated from the following file:

- std_mutex.h

6.183 std::aligned_storage< _Len, _Align > Struct Template Reference

`#include <type_traits>`

6.183.1 Detailed Description

template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
struct std::aligned_storage< _Len, _Align >

Aligned storage.
The member typedef `type` is be a POD type suitable for use as uninitialized storage for any object whose size is at most `_Len` and whose alignment is a divisor of `_Align`.
It is important to use the nested `type` as uninitialized storage, not the `std::aligned_storage` type itself which is an empty class with 1-byte alignment. So this is correct:
`typename std::aligned_storage<sizeof(X), alignof(X)>::type m_xobj;`
This is wrong:
`std::aligned_storage<sizeof(X), alignof(X)> m_xobj;`
In C++14 and later `std::aligned_storage_t<sizeof(X), alignof(X)>` can be used to refer to the `type` member typedef.
The default value of _Align is supposed to be the most stringent fundamental alignment requirement for any C++ object type whose size is no greater than `_Len` (see [basic.align] in the C++ standard).

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

The documentation for this struct was generated from the following file:

- type_traits

6.184 std::aligned_union< _Len, _Types > Struct Template Reference

`#include <type_traits>`

Public Types

- using type

Static Public Attributes

- static const size_t alignment_value

6.184.1 Detailed Description

template<size_t _Len, typename... _Types>
struct std::aligned_union< _Len, _Types >

Provide aligned storage for types.
[meta.trans.other]
Provides aligned storage for any of the provided types of at least size _Len.

See also

aligned_storage

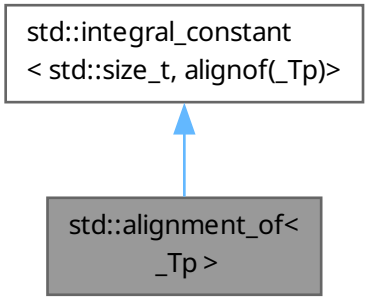
Deprecated Deprecated in C++23.

The documentation for this struct was generated from the following file:

- type_traits

6.185 std::alignment_of< _Tp > Struct Template Reference

`#include <type_traits>`
Inheritance diagram for std::alignment_of< _Tp >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr std::size_t **value**

6.185.1 Detailed Description

template<**typename _Tp**>
struct std::alignment_of< **_Tp** >

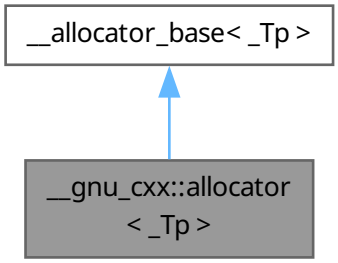
alignment_of
The documentation for this struct was generated from the following file:

- type_traits

6.186 __gnu_cxx::allocator< _Tp > Class Template Reference

`#include <memory>`

Inheritance diagram for __gnu_cxx::allocator< _Tp >:



Public Types

- typedef ptrdiff_t **difference_type**
- using **propagate_on_container_move_assignment**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **allocator** (const allocator &__a) noexcept
- template<typename _Tp1>
  constexpr **allocator** (const allocator< _Tp1 > &) noexcept
- constexpr _Tp * **allocate** (size_t __n)
- constexpr _Tp * **allocate** (size_type __n, const void *=static_cast< const void * >(0))
- constexpr allocation_result< _Tp *, size_t > **allocate_at_least** (size_t __n)
- constexpr void **deallocate** (_Tp *__p, size_t __n)
- allocator & **operator=** (const allocator &)=default

Static Public Member Functions

- static constexpr void **_S_check_allocation_limit** (size_t __n)

Friends

- constexpr bool **operator==** (const allocator &, const allocator &) noexcept

Related Symbols

(Note that these are not member symbols.)

- template<typename _T1, typename _T2>
  constexpr bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept

6.186.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::allocator< _Tp >

The *standard* allocator, as per C++03 [20.4.1].
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/memory.html#std.util.memory.allocator for further details.

Template Parameters

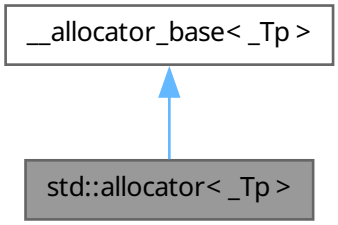
_Tp	Type of allocated object.

The documentation for this class was generated from the following file:

- allocator.h

6.187 std::allocator< _Tp > Class Template Reference

`#include <memory>`
Inheritance diagram for std::allocator< _Tp >:



Public Types

- typedef ptrdiff_t **difference_type**
- using **propagate_on_container_move_assignment**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **allocator** (const allocator &__a) noexcept
- template<typename _Tp1>
  constexpr **allocator** (const allocator< _Tp1 > &) noexcept
- constexpr _Tp * **allocate** (size_type __n, const void *=static_cast< const void * >(0))
- constexpr _Tp * **allocate** (size_t __n)
- constexpr allocation_result< _Tp *, size_t > **allocate_at_least** (size_t __n)
- constexpr void **deallocate** (_Tp *__p, size_t __n)
- allocator & **operator=** (const allocator &)=default

Static Public Member Functions

- static constexpr void **_S_check_allocation_limit** (size_t __n)

Friends

- constexpr bool **operator==** (const allocator &, const allocator &) noexcept

Related Symbols

(Note that these are not member symbols.)

- template<typename _T1, typename _T2>
  constexpr bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept

6.187.1 Detailed Description

template<typename _Tp>
class std::allocator< _Tp >

The *standard* allocator, as per C++03 [20.4.1].
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/memory.html#std.util.memory.allocator for further details.

Template Parameters

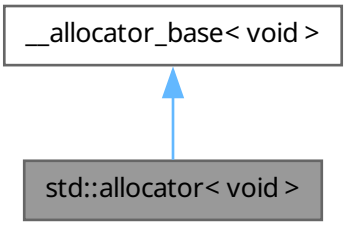
_Tp	Type of allocated object.

The documentation for this class was generated from the following file:

- allocator.h

6.188 std::allocator< void > Class Reference

`#include <memory>`
Inheritance diagram for std::allocator< void >:



Public Types

- typedef ptrdiff_t **difference_type**

- typedef ptrdiff_t **difference_type**
- using **propagate_on_container_move_assignment**
- using **propagate_on_container_move_assignment**
- typedef size_t **size_type**
- typedef size_t **size_type**
- typedef void **value_type**
- typedef void **value_type**

Public Member Functions

- constexpr **allocator** (const allocator &__a) noexcept
- constexpr **allocator** (const allocator< _Tp1 > &) noexcept
- template<typename _Up>
  constexpr **allocator** (const allocator< _Up > &) noexcept
- constexpr void * **allocate** (size_t __n)
- constexpr void * **allocate** (size_type __n, const void *=static_cast< const void * >(0))
- constexpr void * **allocate** (size_type __n, const void *=static_cast< const void * >(0))
- constexpr allocation_result< void *, size_t > **allocate_at_least** (size_t __n)
- constexpr void **deallocate** (_Tp *__p, size_type __n)
- constexpr void **deallocate** (void *__p, size_t __n)
- allocator & **operator=** (const allocator &)=default

Static Public Member Functions

- static constexpr void **_S_check_allocation_limit** (size_t __n)
- static constexpr void **_S_check_allocation_limit** (size_t __n)

Friends

- constexpr bool **operator==** (const allocator &, const allocator &) noexcept

Related Symbols

(Note that these are not member symbols.)

- constexpr bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept
- constexpr bool operator== (const allocator< _T1 > &, const allocator< _T2 > &) noexcept

6.188.1 Detailed Description

std::allocator<void> specialization.

6.188.2 Friends And Related Symbol Documentation

operator==() [1/2]

```
bool operator== (
            const allocator< _T1 > & ,
            const allocator< _T2 > & )  [related]
```

Equality comparison for std::allocator objects

Returns

true, for all std::allocator objects.

operator==() [2/2]

```
bool operator== (
            const allocator< _T1 > & ,
            const allocator< _T2 > & )  [related]
```

Equality comparison for std::allocator objects

Returns

true, for all std::allocator objects.

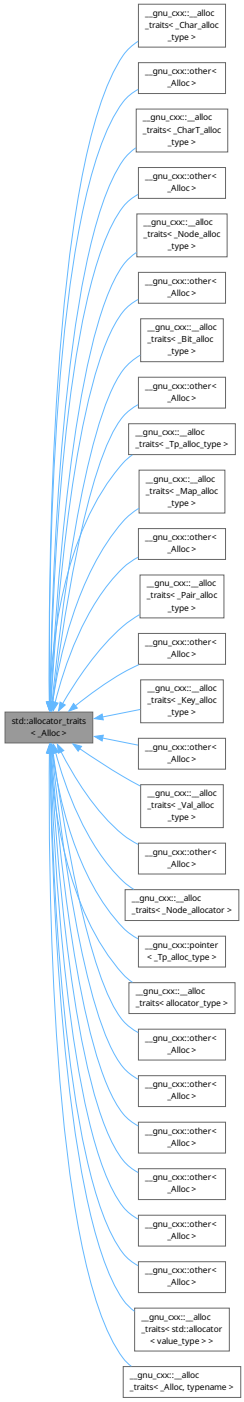
The documentation for this class was generated from the following file:

- allocator.h

6.189 std::allocator_traits< _Alloc > Struct Template Reference

```
#include <memory>
```

Inheritance diagram for std::allocator_traits< _Alloc >:



Public Types

- typedef _Alloc allocator_type
- using const_pointer

- using const_void_pointer
- using difference_type
- using is_always_equal
- using pointer
- using propagate_on_container_copy_assignment
- using propagate_on_container_move_assignment
- using propagate_on_container_swap
- template<typename _Tp>
  using **rebind_alloc**
- template<typename _Tp>
  using **rebind_traits**
- using size_type
- typedef _Alloc::value_type value_type
- using void_pointer

Static Public Member Functions

- static constexpr pointer allocate (_Alloc &__a, size_type __n)
- static constexpr pointer allocate (_Alloc &__a, size_type __n, const_void_pointer __hint)
- static constexpr allocation_result< pointer, size_type > allocate_at_least (_Alloc &__a, size_type __n)
- template<typename _Tp, typename... _Args>
  static constexpr void construct (_Alloc &__a, _Tp *__p, _Args &&... __args) noexcept(_S_nothrow_construct< _Tp, _Args... >())
- static constexpr void deallocate (_Alloc &__a, pointer __p, size_type __n)
- template<typename _Tp>
  static constexpr void destroy (_Alloc &__a, _Tp *__p) noexcept(_S_nothrow_destroy< _Tp >())
- static constexpr size_type max_size (const _Alloc &__a) noexcept
- static constexpr _Alloc select_on_container_copy_construction (const _Alloc &__rhs)

6.189.1 Detailed Description

template<typename _Alloc>
struct std::allocator_traits< _Alloc >

Uniform interface to all allocator types.

Since

C++11

6.189.2 Member Typedef Documentation

allocator_type

```
template<typename _Alloc>
typedef _Alloc std::allocator_traits< _Alloc >::allocator_type
```

The allocator type.

const_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_pointer
```

The allocator's const pointer type.
`Alloc::const_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const value_type>`

const_void_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::const_void_pointer
```

The allocator's const void pointer type.
`Alloc::const_void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<const void>`

difference_type

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::difference_type
```

The allocator's difference type.
`Alloc::difference_type` if that type exists, otherwise `pointer_traits<pointer>::difference_type`

is_always_equal

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::is_always_equal
```

Whether all instances of the allocator type compare equal.
`Alloc::is_always_equal` if that type exists, otherwise `is_empty<Alloc>::type`

pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::pointer
```

The allocator's pointer type.
`Alloc::pointer` if that type exists, otherwise `value_type*`

propagate_on_container_copy_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_copy_assignment
```

How the allocator is propagated on copy assignment.
`Alloc::propagate_on_container_copy_assignment` if that type exists, otherwise `false_type`

propagate_on_container_move_assignment

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_move_assignment
```

How the allocator is propagated on move assignment.
`Alloc::propagate_on_container_move_assignment` if that type exists, otherwise `false_type`

propagate_on_container_swap

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::propagate_on_container_swap
```

How the allocator is propagated on swap.
`Alloc::propagate_on_container_swap` if that type exists, otherwise `false_type`

size_type

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::size_type
```

The allocator's size type.

`Alloc::size_type` if that type exists, otherwise `make_unsigned<difference_type>::type`

value_type

```
template<typename _Alloc>
typedef _Alloc::value_type std::allocator_traits< _Alloc >::value_type
```

The allocated type.

void_pointer

```
template<typename _Alloc>
using std::allocator_traits< _Alloc >::void_pointer
```

The allocator's void pointer type.
`Alloc::void_pointer` if that type exists, otherwise `pointer_traits<pointer>::rebind<void>`

6.189.3 Member Function Documentation

allocate() [1/2]

```
template<typename _Alloc>
constexpr pointer std::allocator_traits< _Alloc >::allocate (
            _Alloc & __a,
            size_type __n)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

_↩ _a	An allocator.
_↩ _n	The number of objects to allocate space for.

Calls `a.allocate(n)`

allocate() [2/2]

```
template<typename _Alloc>
constexpr pointer std::allocator_traits< _Alloc >::allocate (
            _Alloc & __a,
            size_type __n,
            const_void_pointer __hint)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)` if that expression is well-formed, otherwise returns `a.allocate(n)`

allocate_at_least()

```
template<typename _Alloc>
constexpr allocation_result< pointer, size_type > std::allocator_traits< _Alloc >::allocate_at↩
_least (
            _Alloc & __a,
            size_type __n)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory, generously.

Parameters

_↩ _*a*	An allocator.
_↩ _*n*	The minimum number of objects to allocate space for.

Returns

Memory of suitable size and alignment for `n` or more contiguous objects of type `value_type`.

Returns `a.allocate_at_least(n)` if that expression is well-formed, else `{ a.allocate(n), n }`. When an allocator is obliged to reserve more space than required for the cited `n` objects, it may deliver the extra space to the caller.

construct()

```
template<typename _Alloc>
template<typename _Tp, typename...  _Args>
constexpr void std::allocator_traits< _Alloc >::construct (
            _Alloc & __a,
            _Tp * __p,
            _Args &&...  __args)  [inline], [static], [constexpr], [noexcept]
```

Construct an object of type `_Tp`.

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for Tp
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<Args>(__args)...)` if that expression is well-formed, otherwise uses placement-new to construct an object of type *_Tp* at location *__p* from the arguments *__args*...

deallocate()

```
template<typename _Alloc>
constexpr void std::allocator_traits< _Alloc >::deallocate (
```

```
            _Alloc & __a,
            pointer __p,
            size_type __n)  [inline], [static], [constexpr]
```

Deallocate memory.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the memory to deallocate.
_↩ _n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

destroy()

```
template<typename _Alloc>
template<typename _Tp>
constexpr void std::allocator_traits< _Alloc >::destroy (
            _Alloc & __a,
            _Tp * __p)  [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type *_Tp*.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the object to destroy

Calls `__a.destroy(__p)` if that expression is well-formed, otherwise calls `__p->~_Tp()`

max_size()

```
template<typename _Alloc>
constexpr size_type std::allocator_traits< _Alloc >::max_size (
            const _Alloc & __a)  [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

Parameters

_↩ _a	An allocator.

Returns

`__a.max_size()` or `numeric_limits<size_type>::max()`

Returns `__a.max_size()` if that expression is well-formed, otherwise returns `numeric_limits<size_↩ type>::max()`

select_on_container_copy_construction()

```
template<typename _Alloc>
constexpr _Alloc std::allocator_traits< _Alloc >::select_on_container_copy_construction (
            const _Alloc & __rhs)  [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

Parameters

__rhs	An allocator.

Returns

`__rhs.select_on_container_copy_construction()` or *__rhs*

Returns `__rhs.select_on_container_copy_construction()` if that expression is well-formed, otherwise returns *__rhs*

The documentation for this struct was generated from the following file:

- bits/alloc_traits.h

6.190 std::allocator_traits< allocator< _Tp > > Struct Template Reference

`#include <memory>`

Public Types

- using allocator_type
- using const_pointer
- using const_void_pointer
- using difference_type
- using is_always_equal
- using pointer
- using propagate_on_container_copy_assignment
- using propagate_on_container_move_assignment
- using propagate_on_container_swap
- template<typename _Up>
  using **rebind_alloc**
- template<typename _Up>
  using **rebind_traits**
- using size_type
- using value_type
- using void_pointer

Static Public Member Functions

- static constexpr pointer allocate (allocator_type &__a, size_type __n)
- static constexpr pointer allocate (allocator_type &__a, size_type __n, const_void_pointer __hint)
- static constexpr allocation_result< pointer, size_type > allocate_at_least (allocator_type &__a, size_type __n)
- template<typename _Up, typename... _Args>
  static constexpr void construct (allocator_type &__a, _Up *__p, _Args &&... __args) noexcept(__is_nothrow_↩
  new_constructible< _Up, _Args... >)
- static constexpr void deallocate (allocator_type &__a, pointer __p, size_type __n)

- template<typename _Up>
  static constexpr void destroy (allocator_type &__a, _Up *__p) noexcept(is_nothrow_destructible< _Up >::value)
- static constexpr size_type max_size (const allocator_type &__a) noexcept
- static constexpr allocator_type select_on_container_copy_construction (const allocator_type &__rhs)

6.190.1 Detailed Description

template<typename _Tp>
struct std::allocator_traits< allocator< _Tp > >

Partial specialization for `std::allocator`.

Since

C++11

See also

std::allocator_traits

6.190.2 Member Typedef Documentation

allocator_type

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::allocator_type
```

The allocator type.

const_pointer

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::const_pointer
```

The allocator's const pointer type.

const_void_pointer

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::const_void_pointer
```

The allocator's const void pointer type.

difference_type

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::difference_type
```

The allocator's difference type.

is_always_equal

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::is_always_equal
```

Whether all instances of the allocator type compare equal.

pointer

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::pointer
```

The allocator's pointer type.

propagate_on_container_copy_assignment

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_copy_assignment
```
How the allocator is propagated on copy assignment.

propagate_on_container_move_assignment

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_move_assignment
```
How the allocator is propagated on move assignment.

propagate_on_container_swap

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::propagate_on_container_swap
```
How the allocator is propagated on swap.

size_type

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::size_type
```
The allocator's size type.

value_type

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::value_type
```
The allocated type.

void_pointer

```
template<typename _Tp>
using std::allocator_traits< allocator< _Tp > >::void_pointer
```
The allocator's void pointer type.

6.190.3 Member Function Documentation

allocate() [1/2]

```
template<typename _Tp>
constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
            allocator_type & __a,
            size_type __n)  [inline], [static], [constexpr]
```
Allocate memory.

Parameters

_↩ _a	An allocator.
_↩ _n	The number of objects to allocate space for.

Calls `a.allocate(n)`

allocate() [2/2]

```
template<typename _Tp>
constexpr pointer std::allocator_traits< allocator< _Tp > >::allocate (
            allocator_type & __a,
            size_type __n,
            const_void_pointer __hint)  [inline], [static], [constexpr]
```

Allocate memory.

Parameters

__a	An allocator.
__n	The number of objects to allocate space for.
__hint	Aid to locality.

Returns

Memory of suitable size and alignment for *n* objects of type `value_type`

Returns `a.allocate(n, hint)`

allocate_at_least()

```
template<typename _Tp>
constexpr allocation_result< pointer, size_type > std::allocator_traits< allocator< _Tp > >::↩
allocate_at_least (
            allocator_type & __a,
            size_type __n)  [inline], [static], [nodiscard], [constexpr]
```

Allocate memory, generously.

Parameters

_↩ _a	An allocator.
_↩ _n	The minimum number of objects to allocate space for.

Returns

Memory of suitable size and alignment for `m >= n` contiguous objects of type `value_type`, and `m`.

Returns `a.allocate_at_least(n)`.

construct()

```
template<typename _Tp>
template<typename _Up, typename... _Args>
constexpr void std::allocator_traits< allocator< _Tp > >::construct (
            allocator_type & __a,
            _Up * __p,
            _Args &&... __args)  [inline], [static], [constexpr], [noexcept]
```

Construct an object of type _Up.

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for an object of type _Up.
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<_Args>(__args)...)` in C++11, C++14 and C++17. Changed in C++20 to call `std::construct_at(__p, std::forward<_Args>(__args)...)` instead. References std::_Construct(), and std::forward().

deallocate()

```
template<typename _Tp>
constexpr void std::allocator_traits< allocator< _Tp > >::deallocate (
            allocator_type & __a,
            pointer __p,
            size_type __n)  [inline], [static], [constexpr]
```

Deallocate memory.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the memory to deallocate.
_↩ _n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`

destroy()

```
template<typename _Tp>
template<typename _Up>
constexpr void std::allocator_traits< allocator< _Tp > >::destroy (
            allocator_type & __a,
            _Up * __p)  [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type *_Up*.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the object to destroy

Calls `__a.destroy(__p)`.

max_size()

```
template<typename _Tp>
constexpr size_type std::allocator_traits< allocator< _Tp > >::max_size (
            const allocator_type & __a)  [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

Parameters

_↩ _a	An allocator.

Returns

```
__a.max_size()
```

select_on_container_copy_construction()

```
template<typename _Tp>
constexpr allocator_type std::allocator_traits< allocator< _Tp > >::select_on_container_copy_↩
construction (
            const allocator_type & __rhs)  [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

Parameters

__rhs	An allocator.

Returns

```
__rhs
```

The documentation for this struct was generated from the following file:

- bits/alloc_traits.h

6.191 std::allocator_traits< allocator< void > > Struct Reference

```
#include <memory>
```

Public Types

- using allocator_type
- using const_pointer
- using const_void_pointer
- using difference_type
- using is_always_equal
- using pointer
- using propagate_on_container_copy_assignment
- using propagate_on_container_move_assignment
- using propagate_on_container_swap
- template<typename _Up>
  using **rebind_alloc**

- template<typename _Up>
  using **rebind_traits**
- using size_type
- using value_type
- using void_pointer

Static Public Member Functions

- static void * allocate (allocator_type &, size_type, const void *=nullptr)=delete
- static allocation_result< void *, size_type > **allocate_at_least** (allocator_type &, size_type)=delete
- template<typename _Up, typename... _Args>
  static constexpr void construct (allocator_type &, _Up *__p, _Args &&... __args) noexcept(__is_nothrow_new↩
  _constructible< _Up, _Args... >)
- static void deallocate (allocator_type &, void *, size_type)=delete
- template<typename _Up>
  static constexpr void destroy (allocator_type &, _Up *__p) noexcept(is_nothrow_destructible< _Up >::value)
- static size_type max_size (const allocator_type &)=delete
- static constexpr allocator_type select_on_container_copy_construction (const allocator_type &__rhs)

6.191.1 Detailed Description

Explicit specialization for `std::allocator<void>`.

Since

C++11

See also

std::allocator_traits

6.191.2 Member Typedef Documentation

allocator_type

```
using std::allocator_traits< allocator< void > >::allocator_type
```

The allocator type.

const_pointer

```
using std::allocator_traits< allocator< void > >::const_pointer
```

The allocator's const pointer type.

const_void_pointer

```
using std::allocator_traits< allocator< void > >::const_void_pointer
```

The allocator's const void pointer type.

difference_type

```
using std::allocator_traits< allocator< void > >::difference_type
```

The allocator's difference type.

is_always_equal

```
using std::allocator_traits< allocator< void > >::is_always_equal
```

Whether all instances of the allocator type compare equal.

pointer

```
using std::allocator_traits< allocator< void > >::pointer
```
The allocator's pointer type.

propagate_on_container_copy_assignment

```
using std::allocator_traits< allocator< void > >::propagate_on_container_copy_assignment
```
How the allocator is propagated on copy assignment.

propagate_on_container_move_assignment

```
using std::allocator_traits< allocator< void > >::propagate_on_container_move_assignment
```
How the allocator is propagated on move assignment.

propagate_on_container_swap

```
using std::allocator_traits< allocator< void > >::propagate_on_container_swap
```
How the allocator is propagated on swap.

size_type

```
using std::allocator_traits< allocator< void > >::size_type
```
The allocator's size type.

value_type

```
using std::allocator_traits< allocator< void > >::value_type
```
The allocated type.

void_pointer

```
using std::allocator_traits< allocator< void > >::void_pointer
```
The allocator's void pointer type.

6.191.3 Member Function Documentation

allocate()

```
void * std::allocator_traits< allocator< void > >::allocate (
            allocator_type & ,
            size_type ,
            const void *  = nullptr)  [static], [delete]
```
allocate is ill-formed for allocator<void>

construct()

```
template<typename _Up, typename...  _Args>
constexpr void std::allocator_traits< allocator< void > >::construct (
            allocator_type & ,
            _Up * __p,
            _Args &&...  __args)  [inline], [static], [constexpr], [noexcept]
```
Construct an object of type _Up.

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for an object of type _Up.
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<_Args>(__args)...)` in C++11, C++14 and C++17. Changed in C++20 to call `std::construct_at(__p, std::forward<_Args>(__args)...)` instead.
References std::_Construct(), and std::forward().

deallocate()

```
void std::allocator_traits< allocator< void > >::deallocate (
            allocator_type & ,
            void * ,
            size_type )  [static], [delete]
```

deallocate is ill-formed for allocator<void>

destroy()

```
template<typename _Up>
constexpr void std::allocator_traits< allocator< void > >::destroy (
            allocator_type & ,
            _Up * __p)  [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type _Up.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the object to destroy

Invokes the destructor for *__p.
References std::_Destroy().

max_size()

```
size_type std::allocator_traits< allocator< void > >::max_size (
            const allocator_type & )  [static], [delete]
```

max_size is ill-formed for allocator<void>

select_on_container_copy_construction()

```
constexpr allocator_type std::allocator_traits< allocator< void > >::select_on_container_copy_↩
construction (
            const allocator_type & __rhs)  [inline], [static], [constexpr]
```

Obtain an allocator to use when copying a container.

Parameters

__rhs	An allocator.

Returns

`__rhs`

The documentation for this struct was generated from the following file:

- bits/alloc_traits.h

6.192 std::allocator_traits< pmr::polymorphic_allocator< _Tp > > Struct Template Reference

`#include <memory_resource>`

Public Types

- using allocator_type
- using const_pointer
- using const_void_pointer
- using difference_type
- using is_always_equal
- using pointer
- template<typename _Up>
  using **rebind_alloc**
- template<typename _Up>
  using **rebind_traits**
- using size_type
- using value_type
- using void_pointer

Static Public Member Functions

- static pointer allocate (allocator_type &__a, size_type __n)
- static pointer allocate (allocator_type &__a, size_type __n, const_void_pointer)
- static std::allocation_result< pointer, size_type > allocate_at_least (allocator_type &__a, size_type __n)
- template<typename _Up, typename... _Args>
  static void construct (allocator_type &__a, _Up *__p, _Args &&... __args)
- static void deallocate (allocator_type &__a, pointer __p, size_type __n)
- template<typename _Up>
  static constexpr void destroy (allocator_type &, _Up *__p) noexcept(is_nothrow_destructible< _Up >::value)
- static constexpr size_type max_size (const allocator_type &) noexcept

- using propagate_on_container_copy_assignment
- using propagate_on_container_move_assignment
- using propagate_on_container_swap
- static allocator_type select_on_container_copy_construction (const allocator_type &) noexcept

6.192.1 Detailed Description

template<typename _Tp>
struct std::allocator_traits< pmr::polymorphic_allocator< _Tp > >

Partial specialization for `std::pmr::polymorphic_allocator`.

Since

C++17

6.192.2 Member Typedef Documentation

allocator_type

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocator_type
```

The allocator type.

const_pointer

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::const_pointer
```

The allocator's const pointer type.

const_void_pointer

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::const_void_pointer
```

The allocator's const void pointer type.

difference_type

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::difference_type
```

The allocator's difference type.

is_always_equal

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::is_always_equal
```

Whether all instances of the allocator type compare equal.

pointer

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::pointer
```

The allocator's pointer type.

propagate_on_container_copy_assignment

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_copy_↩
assignment
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

propagate_on_container_move_assignment

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_move_↩
assignment
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

propagate_on_container_swap

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::propagate_on_container_swap
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.

size_type

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::size_type
```

The allocator's size type.

value_type

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::value_type
```

The allocated type.

void_pointer

```
template<typename _Tp>
using std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::void_pointer
```

The allocator's void pointer type.

6.192.3 Member Function Documentation

allocate() [1/2]

```
template<typename _Tp>
pointer std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocate (
            allocator_type & __a,
            size_type __n)  [inline], [static], [nodiscard]
```

Allocate memory.

Parameters

_↩ _a	An allocator.
_↩ _n	The number of objects to allocate space for.

Calls `a.allocate(n)`.

allocate() [2/2]

```
template<typename _Tp>
pointer std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::allocate (
            allocator_type & __a,
```

```
            size_type __n,
            const_void_pointer )  [inline], [static], [nodiscard]
```

Allocate memory.

Parameters

_↩ _a	An allocator.
_↩ _n	The number of objects to allocate space for.

Returns

Memory of suitable size and alignment for `n` objects of type `value_type`.

The third parameter is ignored..
Returns `a.allocate(n)`.

allocate_at_least()

```
template<typename _Tp>
std::allocation_result< pointer, size_type > std::allocator_traits< pmr::polymorphic_allocator<
_Tp > >::allocate_at_least (
            allocator_type & __a,
            size_type __n)  [inline], [static], [nodiscard]
```

Allocate memory.

Parameters

_↩ _a	An allocator.
_↩ _n	The number of objects to allocate space for.

Returns

Memory of suitable size and alignment for `n` objects of type `value_type`.

Just returns `{ a.allocate(n), n }`: `polymorphic_allocator` cannot be extended without breaking ABI.

construct()

```
template<typename _Tp>
template<typename _Up, typename...  _Args>
void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::construct (
            allocator_type & __a,
            _Up * __p,
            _Args &&...  __args)  [inline], [static]
```

Construct an object of type `_Up`.

Parameters

__a	An allocator.
__p	Pointer to memory of suitable size and alignment for an object of type `_Up`.
__args	Constructor arguments.

Calls `__a.construct(__p, std::forward<_Args>(__args)...)` in C++11, C++14 and C++17. Changed in C++20 to call `std::construct_at(__p, std::forward<_Args>(__args)...)` instead. References std::forward().

deallocate()

```
template<typename _Tp>
void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::deallocate (
            allocator_type & __a,
            pointer __p,
            size_type __n)  [inline], [static]
```

Deallocate memory.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the memory to deallocate.
_↩ _n	The number of objects space was allocated for.

Calls `a.deallocate(p, n)`.

destroy()

```
template<typename _Tp>
template<typename _Up>
constexpr void std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::destroy (
            allocator_type & ,
            _Up * __p)  [inline], [static], [constexpr], [noexcept]
```

Destroy an object of type `_Up`.

Parameters

_↩ _a	An allocator.
_↩ _p	Pointer to the object to destroy

Calls `p->_Up()`.

max_size()

```
template<typename _Tp>
constexpr size_type std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::max_size (
            const allocator_type & )  [inline], [static], [constexpr], [noexcept]
```

The maximum supported allocation size.

Returns

```
numeric_limits<size_t>::max() / sizeof(value_type)
```

select_on_container_copy_construction()

```
template<typename _Tp>
allocator_type std::allocator_traits< pmr::polymorphic_allocator< _Tp > >::select_on_container↩
_copy_construction (
            const allocator_type & )  [inline], [static], [noexcept]
```

A `polymorphic_allocator` does not propagate when a container is copied, moved, or swapped.
The documentation for this struct was generated from the following file:

- memory_resource.h

6.193 __gnu_cxx::limit_condition::always_adjustor Struct Reference

`#include <throw_allocator.h>`

6.193.1 Detailed Description

Always enter the condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.194 __gnu_cxx::random_condition::always_adjustor Struct Reference

`#include <throw_allocator.h>`

6.194.1 Detailed Description

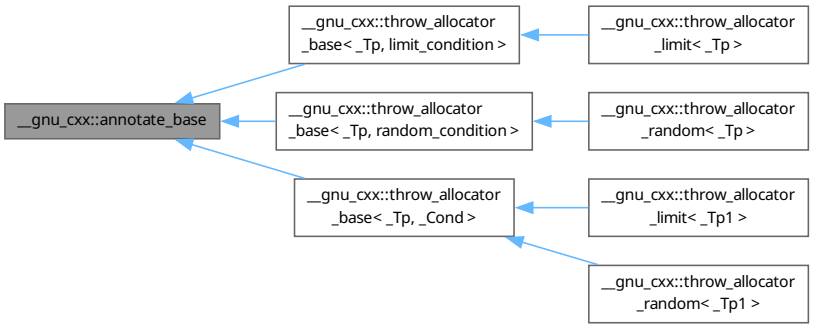
Always enter the condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.195 __gnu_cxx::annotate_base Struct Reference

`#include <throw_allocator.h>`

Inheritance diagram for __gnu_cxx::annotate_base:



Public Member Functions

- void **check** (size_t label)
- map_alloc_type::iterator **check_allocated** (void ∗p, size_t size)
- void **check_constructed** (size_t label)
- map_construct_type::iterator **check_constructed** (void ∗p)
- void **erase** (void ∗p, size_t size)
- void **erase_construct** (void ∗p)
- void **insert** (void ∗p, size_t size)
- void **insert_construct** (void ∗p)

Static Public Member Functions

- static void **check** ()
- static size_t **get_label** ()
- static void **set_label** (size_t l)

Friends

- std::ostream & **operator**<< (std::ostream &os, const annotate_base &__b)

6.195.1 Detailed Description

Base class for checking address and label information about allocations. Create a std::map between the allocated address (void∗) and a datum for annotations, which are a pair of numbers corresponding to label and allocated size.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.196 std::experimental::fundamentals_v1::any Class Reference

```
#include <any>
```

Public Member Functions

- any () noexcept
- template<typename _ValueType, typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>, typename enable_if<!is_constructible< _Tp, _ValueType && >::value, bool >::type = false>
  any (_ValueType &&__value)
- template<typename _ValueType, typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>, typename enable_if< is_constructible< _Tp, _ValueType && >::value, bool >::type = true>
  any (_ValueType &&__value)
- any (any &&__other) noexcept
- any (const any &__other)
- ~any ()
- void clear () noexcept
- bool empty () const noexcept
- template<typename _ValueType>
  enable_if_t<!is_same< any, decay_t< _ValueType > >::value, any & > operator= (_ValueType &&__rhs)
- any & operator= (any &&__rhs) noexcept
- any & operator= (const any &__rhs)
- void swap (any &__rhs) noexcept
- const type_info & type () const noexcept

Static Public Member Functions

- template<typename _Tp>
  static constexpr bool **__is_valid_cast** ()

Friends

- template<typename _Tp>
  enable_if_t< is_object< _Tp >::value, void * > **__any_caster** (const any *__any)

6.196.1 Detailed Description

A type-safe container of any type.
An `any` object's state is either empty or it stores a contained object of CopyConstructible type.

6.196.2 Constructor & Destructor Documentation

any() [1/5]

```
std::experimental::fundamentals_v1::any::any ()  [inline], [noexcept]
```

Default constructor, creates an empty object.
Referenced by any(), any(), operator=(), operator=(), operator=(), and swap().

any() [2/5]

```
std::experimental::fundamentals_v1::any::any (
            const any & __other)  [inline]
```

Copy constructor, copies the state of __other.
References any(), and empty().

any() [3/5]

```
std::experimental::fundamentals_v1::any::any (
            any && __other)  [inline], [noexcept]
```

Move constructor, transfer the state from __other.

Postcondition

__other.empty() (this postcondition is a GNU extension)

References any().

any() [4/5]

```
template<typename _ValueType, typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>,
typename enable_if< is_constructible< _Tp, _ValueType && >::value, bool >::type = true>
std::experimental::fundamentals_v1::any::any (
            _ValueType && __value)  [inline]
```

Construct with a copy of __value as the contained object.
References std::forward().

any() [5/5]

```
template<typename _ValueType, typename _Tp = _Decay<_ValueType>, typename _Mgr = _Manager<_Tp>,
typename enable_if<!is_constructible< _Tp, _ValueType && >::value, bool >::type = false>
std::experimental::fundamentals_v1::any::any (
            _ValueType && __value)  [inline]
```

Construct with a copy of __value as the contained object.

~any()

```
std::experimental::fundamentals_v1::any::~any ()  [inline]
```

Destructor, calls clear().
References clear().

6.196.3 Member Function Documentation

clear()

```
void std::experimental::fundamentals_v1::any::clear ()  [inline], [noexcept]
```

If not empty, destroy the contained object.
References empty().
Referenced by ~any(), and operator=().

empty()

```
bool std::experimental::fundamentals_v1::any::empty () const  [inline], [nodiscard], [noexcept]
```

Reports whether there is a contained object or not.
Referenced by any(), clear(), swap(), and type().

operator=() [1/3]

```
template<typename _ValueType>
enable_if_t<!is_same< any, decay_t< _ValueType > >::value, any & > std::experimental::fundamentals↩
_v1::any::operator= (
            _ValueType && __rhs)  [inline]
```

Store a copy of __rhs as the contained object.
References any(), and std::forward().

operator=() [2/3]

```
any & std::experimental::fundamentals_v1::any::operator= (
            any && __rhs)  [inline], [noexcept]
```

Move assignment operator.

Postcondition

`__rhs.empty()` (not guaranteed for other implementations)

References any(), and clear().

operator=() [3/3]

```
any & std::experimental::fundamentals_v1::any::operator= (
            const any & __rhs)  [inline]
```

Copy the state of another object.
References any().

swap()

```
void std::experimental::fundamentals_v1::any::swap (
            any & __rhs)  [inline], [noexcept]
```

Exchange state with another object.
References any(), and empty().

type()

```
const type_info & std::experimental::fundamentals_v1::any::type () const  [inline], [noexcept]
```

The `typeid` of the contained object, or `typeid(void)` if empty.
References empty().
The documentation for this class was generated from the following file:

- experimental/any

6.197 std::array< _Tp, _Nm > Struct Template Reference

`#include <array>`

Public Types

- typedef const value_type * **const_iterator**
- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef std::ptrdiff_t **difference_type**
- typedef value_type * **iterator**
- typedef value_type * **pointer**
- typedef value_type & **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr reference **at** (size_type __n)
- constexpr const_reference **at** (size_type __n) const
- constexpr const_reference **back** () const noexcept
- constexpr reference **back** () noexcept
- constexpr const_iterator **begin** () const noexcept
- constexpr iterator **begin** () noexcept
- constexpr const_iterator **cbegin** () const noexcept
- constexpr const_iterator **cend** () const noexcept
- constexpr const_reverse_iterator **crbegin** () const noexcept
- constexpr const_reverse_iterator **crend** () const noexcept
- constexpr const_pointer **data** () const noexcept
- constexpr pointer **data** () noexcept
- constexpr bool **empty** () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr iterator **end** () noexcept
- constexpr void **fill** (const value_type &__u)
- constexpr const_reference **front** () const noexcept
- constexpr reference **front** () noexcept
- constexpr size_type **max_size** () const noexcept
- constexpr const_reference **operator[ ]** (size_type __n) const noexcept
- constexpr reference **operator[ ]** (size_type __n) noexcept
- constexpr const_reverse_iterator **rbegin** () const noexcept
- constexpr reverse_iterator **rbegin** () noexcept
- constexpr const_reverse_iterator **rend** () const noexcept
- constexpr reverse_iterator **rend** () noexcept
- constexpr size_type **size** () const noexcept
- constexpr void **swap** (array &__other) noexcept(__array_traits< _Tp, _Nm >::_Is_nothrow_swappable::value)

Public Attributes

- __array_traits< _Tp, _Nm >::_Type **_M_elems**

6.197.1 Detailed Description

template<typename _Tp, std::size_t _Nm>
struct std::array< _Tp, _Nm >

A standard container for storing a fixed size sequence of elements.
Meets the requirements of a container, a reversible container, and a sequence.
Sets support random access iterators.

Template Parameters

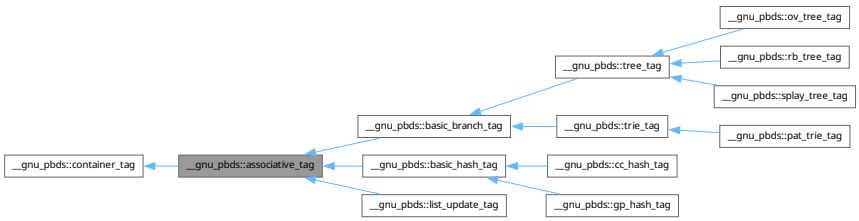
Tp	Type of element. Required to be a complete type.
Nm	Number of elements.

The documentation for this struct was generated from the following file:

- array

6.198 __gnu_pbds::associative_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::associative_tag:



6.198.1 Detailed Description

Basic associative-container.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.199 std::atomic< _Tp > Struct Template Reference

`#include <atomic>`

Public Types

- using **value_type**

Public Member Functions

- constexpr **atomic** (_Tp __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_strong** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) volatile noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (_Tp &__e, _Tp __i, memory_order __s, memory_order __f) volatile noexcept
- _Tp **exchange** (_Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- _Tp **exchange** (_Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatile noexcept
- _Tp **load** (memory_order __m=memory_order_seq_cst) const noexcept
- _Tp **load** (memory_order __m=memory_order_seq_cst) const volatile noexcept
- **operator _Tp** () const noexcept
- **operator _Tp** () const volatile noexcept
- _Tp **operator=** (_Tp __i) noexcept

- _Tp **operator=** (_Tp __i) volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (_Tp __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (_Tp __i, memory_order __m=memory_order_seq_cst) volatile noexcept

6.199.1 Detailed Description

template<typename _Tp>
struct std::atomic< _Tp >

Generic atomic type, primary class template.

Template Parameters

_Tp	Type to be made atomic, must be trivially copyable.

The documentation for this struct was generated from the following file:

- atomic

6.200 std::atomic< _Tp * > Struct Template Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< _Tp * > **__base_type**
- typedef _Tp * **__pointer_type**
- using **difference_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__pointer_type __p) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatile noexcept
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **compare_exchange_weak** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (__pointer_type &__p1, __pointer_type __p2, memory_order __m1, memory_order __m2) volatile noexcept
- bool **compare_exchange_weak** (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (__pointer_type &__p1, __pointer_type __p2, memory_order __m=memory_order_seq_cst) volatile noexcept
- __pointer_type **exchange** (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **exchange** (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept

- __pointer_type **fetch_add** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **fetch_add** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept
- __pointer_type **fetch_max** (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **fetch_max** (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept
- __pointer_type **fetch_min** (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **fetch_min** (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept
- __pointer_type **fetch_sub** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) noexcept
- __pointer_type **fetch_sub** (ptrdiff_t __d, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatile noexcept
- __pointer_type **load** (memory_order __m=memory_order_seq_cst) const noexcept
- __pointer_type **load** (memory_order __m=memory_order_seq_cst) const volatile noexcept
- **operator __pointer_type** () const noexcept
- **operator __pointer_type** () const volatile noexcept
- **operator _Tp** () const noexcept
- __pointer_type **operator++** () noexcept
- __pointer_type **operator++** () volatile noexcept
- __pointer_type **operator++** (int) noexcept
- __pointer_type **operator++** (int) volatile noexcept
- __pointer_type **operator+=** (ptrdiff_t __d) noexcept
- __pointer_type **operator+=** (ptrdiff_t __d) volatile noexcept
- __pointer_type **operator--** () noexcept
- __pointer_type **operator--** () volatile noexcept
- __pointer_type **operator--** (int) noexcept
- __pointer_type **operator--** (int) volatile noexcept
- __pointer_type **operator-=** (ptrdiff_t __d) noexcept
- __pointer_type **operator-=** (ptrdiff_t __d) volatile noexcept
- __pointer_type **operator=** (__pointer_type __p) noexcept
- __pointer_type **operator=** (__pointer_type __p) volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (__pointer_type __p, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (__pointer_type __p, memory_order __m=memory_order_seq_cst) volatile noexcept

Public Attributes

- __base_type **_M_b**

6.200.1 Detailed Description

template<**typename _Tp**>
struct std::atomic< **_Tp** * >

Partial specialization for pointer types.
The documentation for this struct was generated from the following file:

- atomic

6.201 std::atomic< bool > Struct Reference

`#include <atomic>`

Public Types

- using **value_type**

Public Member Functions

- constexpr **atomic** (bool __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2) volatile noexcept
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_strong** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2) noexcept
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m1, memory_order __m2) volatile noexcept
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst) noexcept
- bool **compare_exchange_weak** (bool &__i1, bool __i2, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **exchange** (bool __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **exchange** (bool __i, memory_order __m=memory_order_seq_cst) volatile noexcept
- bool **is_lock_free** () const noexcept
- bool **is_lock_free** () const volatile noexcept
- bool **load** (memory_order __m=memory_order_seq_cst) const noexcept
- bool **load** (memory_order __m=memory_order_seq_cst) const volatile noexcept
- **operator bool** () const noexcept
- **operator bool** () const noexcept
- **operator bool** () const volatile noexcept
- **operator bool** () const volatile noexcept
- bool **operator=** (bool __i) noexcept
- bool **operator=** (bool __i) volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (bool __i, memory_order __m=memory_order_seq_cst) noexcept
- void **store** (bool __i, memory_order __m=memory_order_seq_cst) volatile noexcept

6.201.1 Detailed Description

atomic<bool>
The documentation for this struct was generated from the following file:

- atomic

6.202 std::atomic< char > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< char > **__base_type**
- typedef char **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (char &__e, char __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (char &__e, char __i, memory_order __s, memory_order __f) noexcept
- char **exchange** (char __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept
- char **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator char** () const noexcept
- **operator char** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (char __i, memory_order __m=memory_order_seq_cst) noexcept

6.202.1 Detailed Description

Explicit specialization for char.
The documentation for this struct was generated from the following file:

- atomic

6.203 std::atomic< char16_t > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< char16_t > **__base_type**
- typedef char16_t **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (char16_t &__e, char16_t __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (char16_t &__e, char16_t __i, memory_order __s, memory_order __f) noexcept
- char16_t **exchange** (char16_t __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept
- char16_t **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator char16_t** () const noexcept
- **operator char16_t** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (char16_t __i, memory_order __m=memory_order_seq_cst) noexcept

6.203.1 Detailed Description

Explicit specialization for char16_t.
The documentation for this struct was generated from the following file:

- atomic

6.204 std::atomic< char32_t > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< char32_t > **__base_type**
- typedef char32_t **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (char32_t &__e, char32_t __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (char32_t &__e, char32_t __i, memory_order __s, memory_order __f) noexcept
- char32_t **exchange** (char32_t __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept
- char32_t **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator char32_t** () const noexcept
- **operator char32_t** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (char32_t __i, memory_order __m=memory_order_seq_cst) noexcept

6.204.1 Detailed Description

Explicit specialization for char32_t.
The documentation for this struct was generated from the following file:

- atomic

6.205 std::atomic< int > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< int > **__base_type**
- typedef int **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (int &__e, int __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (int &__e, int __i, memory_order __s, memory_order __f) noexcept
- int **exchange** (int __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept
- int **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator int** () const noexcept
- **operator int** () const volatile noexcept

- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (int __i, memory_order __m=memory_order_seq_cst) noexcept

6.205.1 Detailed Description

Explicit specialization for int.
The documentation for this struct was generated from the following file:

- atomic

6.206 std::atomic< long > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< long > **__base_type**
- typedef long **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (long &__e, long __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (long &__e, long __i, memory_order __s, memory_order __f) noexcept
- long **exchange** (long __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept
- long **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator long** () const noexcept
- **operator long** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (long __i, memory_order __m=memory_order_seq_cst) noexcept

6.206.1 Detailed Description

Explicit specialization for long.
The documentation for this struct was generated from the following file:

- atomic

6.207 std::atomic< long long > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< long long > **__base_type**
- typedef long long **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (long long &__e, long long __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (long long &__e, long long __i, memory_order __s, memory_order __f) noexcept
- long long **exchange** (long long __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept
- long long **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator long long** () const noexcept
- **operator long long** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (long long __i, memory_order __m=memory_order_seq_cst) noexcept

6.207.1 Detailed Description

Explicit specialization for long long.
The documentation for this struct was generated from the following file:

- atomic

6.208 std::atomic< short > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< short > **__base_type**
- typedef short **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (short &__e, short __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (short &__e, short __i, memory_order __s, memory_order __f) noexcept
- short **exchange** (short __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept
- short **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator short** () const noexcept
- **operator short** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (short __i, memory_order __m=memory_order_seq_cst) noexcept

6.208.1 Detailed Description

Explicit specialization for short.
The documentation for this struct was generated from the following file:

- atomic

6.209 std::atomic< signed char > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< signed char > **__base_type**
- typedef signed char **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (signed char &__e, signed char __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (signed char &__e, signed char __i, memory_order __s, memory_order __f) noexcept
- signed char **exchange** (signed char __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept
- signed char **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator signed char** () const noexcept
- **operator signed char** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (signed char __i, memory_order __m=memory_order_seq_cst) noexcept

6.209.1 Detailed Description

Explicit specialization for signed char.
The documentation for this struct was generated from the following file:

- atomic

6.210 std::atomic< unsigned char > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< unsigned char > **__base_type**
- typedef unsigned char **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (unsigned char &__e, unsigned char __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (unsigned char &__e, unsigned char __i, memory_order __s, memory_order __f) noexcept
- unsigned char **exchange** (unsigned char __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept

- unsigned char **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator unsigned char** () const noexcept
- **operator unsigned char** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (unsigned char __i, memory_order __m=memory_order_seq_cst) noexcept

6.210.1 Detailed Description

Explicit specialization for unsigned char.
The documentation for this struct was generated from the following file:

- atomic

6.211 std::atomic< unsigned int > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< unsigned int > **__base_type**
- typedef unsigned int **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (unsigned int &__e, unsigned int __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (unsigned int &__e, unsigned int __i, memory_order __s, memory_order __f) noexcept
- unsigned int **exchange** (unsigned int __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept
- unsigned int **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator unsigned int** () const noexcept
- **operator unsigned int** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (unsigned int __i, memory_order __m=memory_order_seq_cst) noexcept

6.211.1 Detailed Description

Explicit specialization for unsigned int.
The documentation for this struct was generated from the following file:

- atomic

6.212 std::atomic< unsigned long > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< unsigned long > **__base_type**
- typedef unsigned long **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (unsigned long &__e, unsigned long __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (unsigned long &__e, unsigned long __i, memory_order __s, memory_order __f) noexcept
- unsigned long **exchange** (unsigned long __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept
- unsigned long **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator unsigned long** () const noexcept
- **operator unsigned long** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (unsigned long __i, memory_order __m=memory_order_seq_cst) noexcept

6.212.1 Detailed Description

Explicit specialization for unsigned long.
The documentation for this struct was generated from the following file:

- atomic

6.213 std::atomic< unsigned long long > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< unsigned long long > **__base_type**
- typedef unsigned long long **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (unsigned long long &__e, unsigned long long __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (unsigned long long &__e, unsigned long long __i, memory_order __s, memory_order __f) noexcept
- unsigned long long **exchange** (unsigned long long __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept
- unsigned long long **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator unsigned long long** () const noexcept
- **operator unsigned long long** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (unsigned long long __i, memory_order __m=memory_order_seq_cst) noexcept

6.213.1 Detailed Description

Explicit specialization for unsigned long long.
The documentation for this struct was generated from the following file:

- atomic

6.214 std::atomic< unsigned short > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< unsigned short > **__base_type**
- typedef unsigned short **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (unsigned short &__e, unsigned short __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (unsigned short &__e, unsigned short __i, memory_order __s, memory_order __f) noexcept
- unsigned short **exchange** (unsigned short __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept
- unsigned short **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator unsigned short** () const noexcept
- **operator unsigned short** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (unsigned short __i, memory_order __m=memory_order_seq_cst) noexcept

6.214.1 Detailed Description

Explicit specialization for unsigned short.
The documentation for this struct was generated from the following file:

- atomic

6.215 std::atomic< wchar_t > Struct Reference

`#include <atomic>`

Public Types

- typedef __atomic_base< wchar_t > **__base_type**
- typedef wchar_t **__integral_type**
- using **value_type**

Public Member Functions

- constexpr **atomic** (__integral_type __i) noexcept
- **atomic** (const atomic &)=delete
- bool **compare_exchange_strong** (wchar_t &__e, wchar_t __i, memory_order __s, memory_order __f) noexcept
- bool **compare_exchange_weak** (wchar_t &__e, wchar_t __i, memory_order __s, memory_order __f) noexcept
- wchar_t **exchange** (wchar_t __i, memory_order __m=memory_order_seq_cst) noexcept
- bool **is_lock_free** () const noexcept
- wchar_t **load** (memory_order __m=memory_order_seq_cst) const noexcept
- **operator wchar_t** () const noexcept
- **operator wchar_t** () const volatile noexcept
- atomic & **operator=** (const atomic &) volatile =delete
- atomic & **operator=** (const atomic &)=delete
- void **store** (wchar_t __i, memory_order __m=memory_order_seq_cst) noexcept

6.215.1 Detailed Description

Explicit specialization for wchar_t.
The documentation for this struct was generated from the following file:

- atomic

6.216 std::atomic_flag Struct Reference

`#include <atomic_base.h>`

Public Member Functions

- constexpr **atomic_flag** (bool __i) noexcept
- **atomic_flag** (const atomic_flag &)=delete
- void **clear** (memory_order __m=memory_order_seq_cst) noexcept
- void **clear** (memory_order __m=memory_order_seq_cst) volatile noexcept
- atomic_flag & **operator=** (const atomic_flag &) volatile =delete
- atomic_flag & **operator=** (const atomic_flag &)=delete
- bool **test** (memory_order __m=memory_order_seq_cst) const noexcept
- bool **test** (memory_order __m=memory_order_seq_cst) const volatile noexcept
- bool **test_and_set** (memory_order __m=memory_order_seq_cst) noexcept
- bool **test_and_set** (memory_order __m=memory_order_seq_cst) volatile noexcept

6.216.1 Detailed Description

atomic_flag
The documentation for this struct was generated from the following file:

- atomic_base.h

6.217 std::auto_ptr< _Tp > Class Template Reference

`#include <memory>`

Public Types

- typedef _Tp element_type

Public Member Functions

- auto_ptr (auto_ptr &__a) throw ()
- template<typename _Tp1>
  auto_ptr (auto_ptr< _Tp1 > &__a) throw ()
- auto_ptr (auto_ptr_ref< element_type > __ref) throw ()
- auto_ptr (element_type *__p=0) throw ()
- ~auto_ptr ()
- element_type * get () const throw ()
- template<typename _Tp1>
  operator auto_ptr< _Tp1 > () throw ()
- template<typename _Tp1>
  operator auto_ptr_ref< _Tp1 > () throw ()
- element_type & operator* () const throw ()
- element_type * operator-> () const throw ()
- auto_ptr & operator= (auto_ptr &__a) throw ()
- template<typename _Tp1>
  auto_ptr & operator= (auto_ptr< _Tp1 > &__a) throw ()
- auto_ptr & **operator=** (auto_ptr_ref< element_type > __ref) throw ()
- element_type * release () throw ()
- void reset (element_type *__p=0) throw ()

6.217.1 Detailed Description

template<typename _Tp>
class std::auto_ptr< _Tp >

A simple smart pointer providing strict ownership semantics.
The Standard says:

```
An  auto_ptr owns the object it holds a pointer to.  Copying
an  auto_ptr copies the pointer and transfers ownership to the
destination.  If more than one  auto_ptr owns the same object
at the same time the behavior of the program is undefined.

The uses of  auto_ptr include providing temporary
exception-safety for dynamically allocated memory, passing
ownership of dynamically allocated memory to a function, and
returning dynamically allocated memory from a function.   auto_ptr does not meet the CopyConstructib
requirements for Standard Library container elements and thus
instantiating a Standard Library container with an  auto_ptr results in undefined behavior.
```

Quoted from [20.4.5]/3.
Good examples of what can and cannot be done with auto_ptr can be found in the libstdc++ testsuite.
_GLIBCXX_RESOLVE_LIB_DEFECTS

1. auto_ptr<> conversion issues These resolutions have all been incorporated.

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

6.217.2 Member Typedef Documentation

element_type

```
template<typename _Tp>
typedef _Tp std::auto_ptr< _Tp >::element_type
```

The pointed-to type.

6.217.3 Constructor & Destructor Documentation

auto_ptr() [1/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
            element_type * __p = 0) throw ( )   [inline], [explicit]
```

An auto_ptr is usually constructed from a raw pointer.

Parameters

_↩ _p	A pointer (defaults to NULL).

This object now *owns* the object pointed to by *__p*.
Referenced by auto_ptr(), auto_ptr(), operator=(), and operator=().

auto_ptr() [2/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
            auto_ptr< _Tp > & __a) throw ( )   [inline]
```

An auto_ptr can be constructed from another auto_ptr.

Parameters

_↩ _a	Another auto_ptr of the same type.

This object now *owns* the object previously owned by *__a*, which has given up ownership.
References auto_ptr().

auto_ptr() [3/4]

```
template<typename _Tp>
template<typename _Tp1>
std::auto_ptr< _Tp >::auto_ptr (
            auto_ptr< _Tp1 > & __a) throw ( )   [inline]
```

An auto_ptr can be constructed from another auto_ptr.

Parameters

_↩ _a	Another auto_ptr of a different but related type.

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element_type.
This object now *owns* the object previously owned by *__a*, which has given up ownership.
References auto_ptr().

~auto_ptr()

```
template<typename _Tp>
std::auto_ptr< _Tp >::~auto_ptr ()  [inline]
```

When the auto_ptr goes out of scope, the object it owns is deleted. If it no longer owns anything (i.e., `get()` is `NULL`), then this has no effect.
The C++ standard says there is supposed to be an empty throw specification here, but omitting it is standard conforming. Its presence can be detected only if _Tp::~_Tp() throws, but this is prohibited. [17.4.3.6]/2

auto_ptr() [4/4]

```
template<typename _Tp>
std::auto_ptr< _Tp >::auto_ptr (
            auto_ptr_ref< element_type > __ref) throw ( )   [inline]
```

Automatic conversions.
These operations are supposed to convert an auto_ptr into and from an auto_ptr_ref automatically as needed. This would allow constructs such as

```
auto_ptr<Derived>  func_returning_auto_ptr(.....);
...
auto_ptr<Base> ptr = func_returning_auto_ptr(.....);
```

But it doesn't work, and won't be fixed. For further details see https://cplusplus.github.io/LWG/↩ lwg-closed.html#463

6.217.4 Member Function Documentation

get()

```
template<typename _Tp>
element_type * std::auto_ptr< _Tp >::get () const throw ( )   [inline]
```

Bypassing the smart pointer.

Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

Note

This auto_ptr still owns the memory.

operator*()

```
template<typename _Tp>
element_type & std::auto_ptr< _Tp >::operator* () const throw ( )   [inline]
```

Smart pointer dereferencing.
If this auto_ptr no longer owns anything, then this operation will crash. (For a smart pointer, *no longer owns anything* is the same as being a null pointer, and you know what happens when you dereference one of those...)

operator->()

```
template<typename _Tp>
element_type * std::auto_ptr< _Tp >::operator-> () const throw ( )   [inline]
```

Smart pointer dereferencing.
This returns the pointer itself, which the language then will automatically cause to be dereferenced.

operator=() [1/2]

```
template<typename _Tp>
auto_ptr & std::auto_ptr< _Tp >::operator= (
            auto_ptr< _Tp > & __a) throw ( )   [inline]
```

auto_ptr assignment operator.

Parameters

_↩ _a	Another auto_ptr of the same type.

This object now *owns* the object previously owned by __a, which has given up ownership. The object that this one *used* to own and track has been deleted.
References auto_ptr(), and reset().

operator=() [2/2]

```
template<typename _Tp>
template<typename _Tp1>
auto_ptr & std::auto_ptr< _Tp >::operator= (
            auto_ptr< _Tp1 > & __a) throw ( )   [inline]
```

auto_ptr assignment operator.

Parameters

_↩ _a	Another auto_ptr of a different but related type.

A pointer-to-Tp1 must be convertible to a pointer-to-Tp/element_type.
This object now *owns* the object previously owned by __a, which has given up ownership. The object that this one *used* to own and track has been deleted.
References auto_ptr(), and reset().

release()

```
template<typename _Tp>
element_type * std::auto_ptr< _Tp >::release () throw ( )   [inline]
```

Bypassing the smart pointer.

Returns

The raw pointer being managed.

You can get a copy of the pointer that this object owns, for situations such as passing to a function which only accepts a raw pointer.

Note

This auto_ptr no longer owns the memory. When this object goes out of scope, nothing will happen.

reset()

```
template<typename _Tp>
void std::auto_ptr< _Tp >::reset (
            element_type * __p = 0) throw ( )   [inline]
```

Forcibly deletes the managed object.

Parameters

_↩ _p	A pointer (defaults to NULL).

This object now *owns* the object pointed to by __*p*. The previous object has been deleted.
Referenced by operator=(), and operator=().
The documentation for this class was generated from the following files:

- shared_ptr_base.h
- auto_ptr.h

6.218 std::auto_ptr_ref< _Tp1 > Struct Template Reference

`#include <auto_ptr.h>`

Public Member Functions

- **auto_ptr_ref** (_Tp1 *__p)

Public Attributes

- _Tp1 * **_M_ptr**

6.218.1 Detailed Description

template<typename _Tp1>
struct std::auto_ptr_ref< _Tp1 >

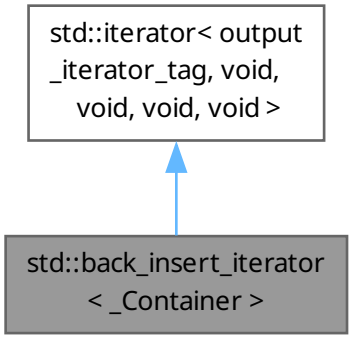
A wrapper class to provide auto_ptr with reference semantics. For example, an auto_ptr can be assigned (or constructed from) the result of a function which returns an auto_ptr by value.
All the auto_ptr_ref stuff should happen behind the scenes.
The documentation for this struct was generated from the following file:

- auto_ptr.h

6.219 std::back_insert_iterator< _Container > Class Template Reference

`#include <stl_iterator.h>`

Inheritance diagram for std::back_insert_iterator< _Container >:



Public Types

- typedef _Container container_type
- using **difference_type**
- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

Public Member Functions

- constexpr back_insert_iterator (_Container &__x)
- constexpr back_insert_iterator & operator* ()
- constexpr back_insert_iterator & operator++ ()
- constexpr back_insert_iterator operator++ (int)
- constexpr back_insert_iterator & operator= (const typename _Container::value_type &__value)
- constexpr back_insert_iterator & **operator=** (typename _Container::value_type &&__value)

Protected Attributes

- _Container * **container**

6.219.1 Detailed Description

template<**typename _Container**>
class std::back_insert_iterator< **_Container** >

Turns assignment into insertion.
These are output iterators, constructed from a container-of-T. Assigning a T to the iterator appends it to the container using push_back.
Tip: Using the back_inserter function to create these iterators can save typing.

6.219.2 Member Typedef Documentation

container_type

```
template<typename _Container>
typedef _Container std::back_insert_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

iterator_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::↩
iterator_category  [inherited]
```

One of the tag types.

pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer  [inherited]
```

This type represents a pointer-to-value_type.

reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference  [inherited]
```

This type represents a reference-to-value_type.

value_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type  [inherited]
```

The type "pointed to" by the iterator.

6.219.3 Constructor & Destructor Documentation

back_insert_iterator()

```
template<typename _Container>
std::back_insert_iterator< _Container >::back_insert_iterator (
            _Container & __x)  [inline], [explicit], [constexpr]
```

The only way to create this iterator is with a container.

References std::__addressof().

6.219.4 Member Function Documentation

operator*()

```
template<typename _Container>
back_insert_iterator & std::back_insert_iterator< _Container >::operator* ()  [inline], [nodiscard],
[constexpr]
```

Simply returns *this.

operator++() [1/2]

```
template<typename _Container>
back_insert_iterator & std::back_insert_iterator< _Container >::operator++ ()  [inline], [constexpr]
```

Simply returns *this. (This iterator does not *move*.).

operator++() [2/2]

```
template<typename _Container>
back_insert_iterator std::back_insert_iterator< _Container >::operator++ (
            int )  [inline], [constexpr]
```

Simply returns ∗this. (This iterator does not *move*.).

operator=()

```
template<typename _Container>
back_insert_iterator & std::back_insert_iterator< _Container >::operator= (
            const typename _Container::value_type & __value)  [inline], [constexpr]
```

Parameters

__value	An instance of whatever type container_type::const_reference is; presumably a reference-to-const T for container<T>.

Returns

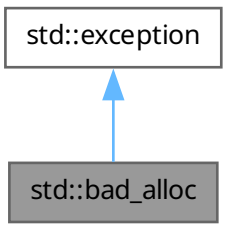
This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the end, if you like). Assigning a value to the iterator will always append the value to the end of the container.
The documentation for this class was generated from the following file:

- bits/stl_iterator.h

6.220 std::bad_alloc Class Reference

`#include <new_except.h>`
Inheritance diagram for std::bad_alloc:



Public Member Functions

- _GLIBCXX26_CONSTEXPR **bad_alloc** (const bad_alloc &)=default
- _GLIBCXX26_CONSTEXPR bad_alloc & **operator=** (const bad_alloc &)=default
- virtual constexpr const char ∗ what () const noexcept

6.220.1 Detailed Description

Exception possibly thrown by `new`.
`bad_alloc` (or classes derived from it) is used to report allocation errors from the throwing forms of `new`.

6.220.2 Member Function Documentation

what()

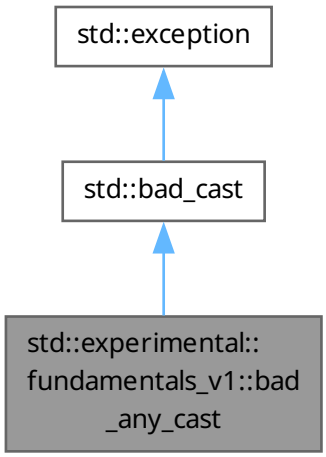
```
virtual constexpr const char * std::bad_alloc::what () const  [inline], [constexpr], [virtual],
[noexcept]
```

Returns a C-style character string describing the general cause of the current error.
Reimplemented from std::exception.
The documentation for this class was generated from the following file:

- new_except.h

6.221 std::experimental::fundamentals_v1::bad_any_cast Class Reference

`#include <any>`
Inheritance diagram for std::experimental::fundamentals_v1::bad_any_cast:



Public Member Functions

- virtual const char * what () const noexcept

6.221.1 Detailed Description

Exception class thrown by a failed `any_cast`.

6.221.2 Member Function Documentation

what()

```
virtual const char * std::experimental::fundamentals_v1::bad_any_cast::what () const  [inline],
[virtual], [noexcept]
```

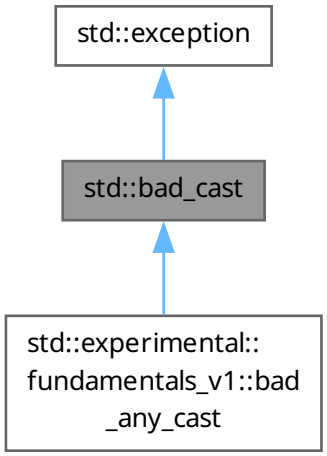
Returns a C-style character string describing the general cause of the current error.
Reimplemented from std::bad_cast.
The documentation for this class was generated from the following file:

- experimental/any

6.222 std::bad_cast Class Reference

```
#include <typeinfo>
```

Inheritance diagram for std::bad_cast:



Public Member Functions

- virtual constexpr const char * what () const noexcept

6.222.1 Detailed Description

Thrown during incorrect typecasting.
If you attempt an invalid `dynamic_cast` expression, an instance of this class (or something derived from this class) is thrown.

6.222.2 Member Function Documentation

what()

```
virtual constexpr const char * std::bad_cast::what () const  [inline], [constexpr], [virtual],
[noexcept]
```

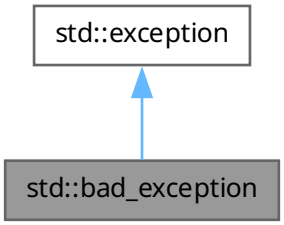
Returns a C-style character string describing the general cause of the current error.

Reimplemented from std::exception.
Reimplemented in std::experimental::fundamentals_v1::bad_any_cast.
The documentation for this class was generated from the following file:

- typeinfo

6.223 std::bad_exception Class Reference

`#include <exception>`
Inheritance diagram for std::bad_exception:



Public Member Functions

- virtual constexpr const char * what () const noexcept

6.223.1 Detailed Description

If an exception is thrown which is not listed in a function's exception specification, one of these may be thrown.

6.223.2 Member Function Documentation

what()

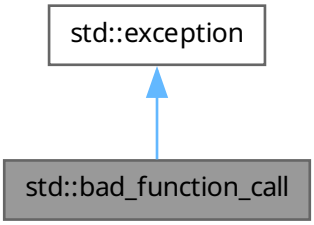
`virtual constexpr const char * std::bad_exception::what () const [inline], [constexpr], [virtual], [noexcept]`
Returns a C-style character string describing the general cause of the current error.
Reimplemented from std::exception.
The documentation for this class was generated from the following file:

- exception

6.224 std::bad_function_call Class Reference

`#include <std_function.h>`

Inheritance diagram for std::bad_function_call:



Public Member Functions

- const char * what () const noexcept

6.224.1 Detailed Description

Exception class thrown when class template function's operator() is called with an empty target.

6.224.2 Member Function Documentation

what()

```
const char * std::bad_function_call::what () const  [virtual], [noexcept]
```

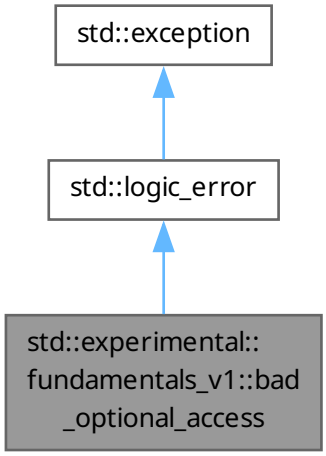
Returns a C-style character string describing the general cause of the current error.
Reimplemented from std::exception.
The documentation for this class was generated from the following file:

- std_function.h

6.225 std::experimental::fundamentals_v1::bad_optional_access Class Reference

```
#include <optional>
```

Inheritance diagram for std::experimental::fundamentals_v1::bad_optional_access:



Public Member Functions

- **bad_optional_access** (const char ∗__arg)
- virtual const char ∗ what () const noexcept

6.225.1 Detailed Description

Exception class thrown when a disengaged optional object is dereferenced.

6.225.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

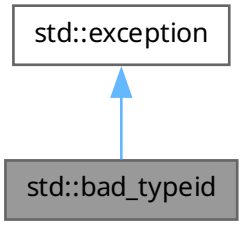
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
References what().
Referenced by what().
The documentation for this class was generated from the following file:

- experimental/optional

6.226 std::bad_typeid Class Reference

```
#include <typeinfo>
```

Inheritance diagram for std::bad_typeid:



Public Member Functions

- virtual constexpr const char * what () const noexcept

6.226.1 Detailed Description

Thrown when a NULL pointer in a `typeid` expression is used.

6.226.2 Member Function Documentation

what()

```
virtual constexpr const char * std::bad_typeid::what () const  [inline], [constexpr], [virtual],
[noexcept]
```

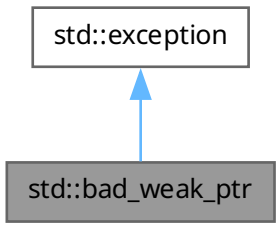
Returns a C-style character string describing the general cause of the current error.
Reimplemented from std::exception.
The documentation for this class was generated from the following file:

- typeinfo

6.227 std::bad_weak_ptr Class Reference

```
#include <shared_ptr_base.h>
```

Inheritance diagram for std::bad_weak_ptr:



Public Member Functions

- virtual char const * what () const noexcept

6.227.1 Detailed Description

Exception possibly thrown by `shared_ptr`.

6.227.2 Member Function Documentation

what()

```
virtual char const * std::bad_weak_ptr::what () const  [virtual], [noexcept]
```

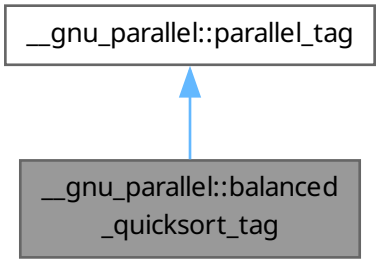
Returns a C-style character string describing the general cause of the current error.
Reimplemented from std::exception.
The documentation for this class was generated from the following file:

- shared_ptr_base.h

6.228 __gnu_parallel::balanced_quicksort_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::balanced_quicksort_tag:



Public Member Functions

- **balanced_quicksort_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

6.228.1 Detailed Description

Forces parallel sorting using balanced quicksort at compile time.

6.228.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

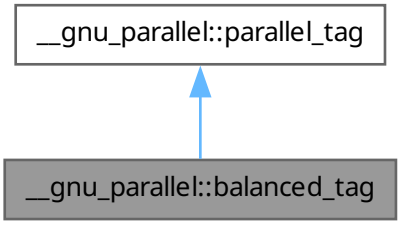
__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

6.229 __gnu_parallel::balanced_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::balanced_tag:



Public Member Functions

- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

6.229.1 Detailed Description

Recommends parallel execution using dynamic load-balancing at compile time.

6.229.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```
Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```
Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

6.230 std::tr2::bases< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- typedef __reflection_typelist< __bases(_Tp)... > **type**

6.230.1 Detailed Description

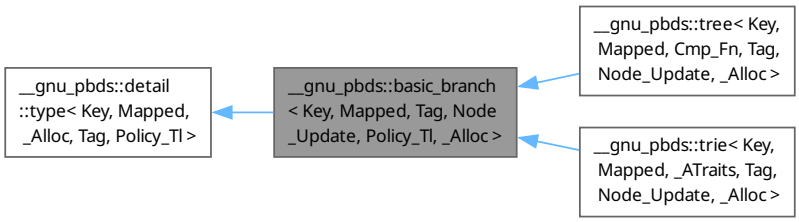
template<typename _Tp>
struct std::tr2::bases< _Tp >

Sequence abstraction metafunctions for manipulating a typelist.
Enumerate all the base classes of a class. Form of a typelist.
The documentation for this struct was generated from the following file:

- tr2/type_traits

6.231 __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc > Class Template Reference

`#include <assoc_container.hpp>`
Inheritance diagram for __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >:



Public Types

- typedef Node_Update **node_update**

Protected Member Functions

- **basic_branch** (const basic_branch &other)
- template<typename T0>
  basic_branch (T0 t0)
- template<typename T0, typename T1>
  basic_branch (T0 t0, T1 t1)
- template<typename T0, typename T1, typename T2>
  basic_branch (T0 t0, T1 t1, T2 t2)
- template<typename T0, typename T1, typename T2, typename T3>
  basic_branch (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0, typename T1, typename T2, typename T3, typename T4>
  basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)

- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5>
  basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>
  basic_branch (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)

6.231.1 Detailed Description

template<typename Key, typename Mapped, typename Tag, typename Node_Update, typename Policy_Tl, typename _Alloc>
class __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >

A branched, tree-like (tree, trie) container abstraction.

Template Parameters

Key	Key type.
Mapped	Map type.
Tag	Instantiating data structure type, see container_tag.
Node_Update	Updates nodes, restores invariants.
Policy_TL	Policy typelist.
_Alloc	Allocator type.

Base is dispatched at compile time via Tag, from the following choices: tree_tag, trie_tag, and their descendants. Base choices are: detail::ov_tree_map, detail::rb_tree_map, detail::splay_tree_map, and detail::pat_trie_map.
The documentation for this class was generated from the following file:

- assoc_container.hpp

6.232 __gnu_pbds::basic_branch_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::basic_branch_tag:



6.232.1 Detailed Description

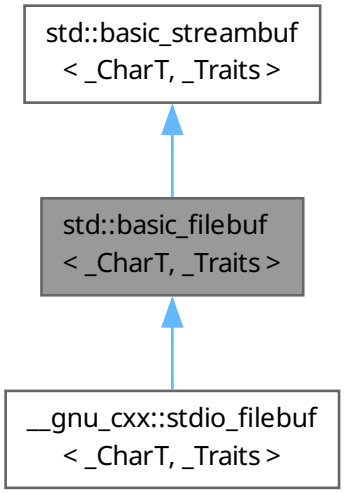
Basic branch structure.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.233 std::basic_filebuf< _CharT, _Traits > Class Template Reference

`#include <fstream>`

Inheritance diagram for std::basic_filebuf< _CharT, _Traits >:



Public Types

- typedef codecvt< char_type, char, __state_type > **__codecvt_type**
- typedef __basic_file< char > **__file_type**
- typedef basic_filebuf< char_type, traits_type > **__filebuf_type**
- typedef traits_type::state_type **__state_type**
- typedef basic_streambuf< char_type, traits_type > **__streambuf_type**
- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**
- typedef _Traits **traits_type**

Public Member Functions

- basic_filebuf ()
- **basic_filebuf** (basic_filebuf &&)
- **basic_filebuf** (const basic_filebuf &)=delete
- virtual ~basic_filebuf ()
- __filebuf_type * close ()
- locale getloc () const
- streamsize in_avail ()
- bool is_open () const throw ()
- template<typename _Path>
  _If_fs_path< _Path, __filebuf_type * > open (const _Path &__s, ios_base::openmode __mode)
- __filebuf_type * open (const char *__s, ios_base::openmode __mode)
- __filebuf_type * open (const std::string &__s, ios_base::openmode __mode)

- basic_filebuf & **operator=** (basic_filebuf &&)
- basic_filebuf & **operator=** (const basic_filebuf &)=delete
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)
- int_type sungetc ()
- void **swap** (basic_filebuf &)

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void **_M_allocate_internal_buffer** ()
- bool **_M_convert_to_external** (char_type *, streamsize)
- void _M_create_pback ()
- void **_M_destroy_internal_buffer** () throw ()
- void _M_destroy_pback () throw ()
- int **_M_get_ext_pos** (__state_type &__state)
- pos_type **_M_seek** (off_type __off, ios_base::seekdir __way, __state_type __state)
- void _M_set_buffer (streamsize __off)
- bool **_M_terminate_output** ()
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=_Traits::eof())
- virtual int_type pbackfail (int_type __c=_Traits::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual __streambuf_type * setbuf (char_type *__s, streamsize __n)
- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)

- char_type * eback () const

- char_type * gptr () const
- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- char_type * _M_buf
- bool **_M_buf_allocated**
- locale _M_buf_locale
- size_t _M_buf_size
- const __codecvt_type * **_M_codecvt**
- char * _M_ext_buf
- streamsize _M_ext_buf_size
- char * **_M_ext_end**
- const char * _M_ext_next
- __file_type **_M_file**
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- __c_lock **_M_lock**
- ios_base::openmode _M_mode
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end
- bool _M_reading
- __state_type **_M_state_beg**
- __state_type **_M_state_cur**
- __state_type **_M_state_last**
- bool **_M_writing**

- char_type _M_pback
- char_type * _M_pback_cur_save
- char_type * _M_pback_end_save
- bool _M_pback_init

Friends

- class **ios_base**

6.233.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_filebuf< _CharT, _Traits >

The actual work of input and output (for files).

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class associates both its input and output sequence with an external disk file, and maintains a joint file position for both sequences. Many of its semantics are described in terms of similar behavior in the Standard C Library's `FILE` streams.
Requirements on traits_type, specific to this class:

- traits_type::pos_type must be fpos<traits_type::state_type>
- traits_type::off_type must be streamoff
- traits_type::state_type must be Assignable and DefaultConstructible,
- traits_type::state_type() must be the initial state for codecvt.

6.233.2 Constructor & Destructor Documentation

basic_filebuf()

```
template<typename _CharT, typename _Traits>
std::basic_filebuf< _CharT, _Traits >::basic_filebuf ()
```

Does not open any files.
The default constructor initializes the parent class using its own default ctor.
References _M_buf, std::basic_streambuf< _CharT, _Traits >::_M_buf_locale, _M_buf_size, _M_ext_buf, _M_ext_buf_size, _M_ext_next, _M_mode, _M_pback, _M_pback_cur_save, _M_pback_end_save, _M_pback_init, and _M_reading.
Referenced by __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), and close().

~basic_filebuf()

```
template<typename _CharT, typename _Traits>
virtual std::basic_filebuf< _CharT, _Traits >::~basic_filebuf ()  [inline], [virtual]
```

The destructor closes the file first.

6.233.3 Member Function Documentation

_M_create_pback()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_create_pback ()  [inline], [protected]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: _M_in_cur has already been moved back
Referenced by pbackfail().

_M_destroy_pback()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback () throw ( )  [inline], [protected]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.
Referenced by overflow(), seekoff(), seekpos(), underflow(), and xsgetn().

_M_set_buffer()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_set_buffer (
            streamsize __off)  [inline], [protected]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:
__off == egptr() - eback() upon underflow/uflow (**read** mode); __off == 0 upon overflow (**write** mode); __off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).
NB: epptr() - pbase() == _M_buf_size - 1, since _M_buf_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.
Referenced by __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), close(), imbue(), open(), overflow(), pbackfail(), underflow(), xsgetn(), and xsputn().

close()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::close
()
```

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.
If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.
If any operations fail, this function also fails.
References basic_filebuf(), _M_mode, _M_pback_init, _M_reading, _M_set_buffer(), and is_open().
Referenced by std::basic_filebuf< char_type, traits_type >::~basic_filebuf(), and open().

eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _Cha std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence

- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_streambuf< char_type, traits_type >::in_avail std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< char std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sputc(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_streambuf< char_type, traits_type >::uflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const  [inline], [inherited]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.
Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy
std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_stringbuf< _CharT,
std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< cha
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_streambuf< char_type, traits_type >::sgetc(),
std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< char >::showmanyc(), std::basic_streambuf< char_type, traits
std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_streambuf< char_type, traits_type >::uflow(),
std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(),
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and
xsgetn().

imbue()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::imbue (
             const locale & __loc)  [protected], [virtual]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_ext_buf, _M_ext_next, _M_mode, _M_reading, _M_set_buffer(), std::ios_base::cur, std::basic_streambuf< _CharT, _Trait
std::basic_streambuf< _CharT, _Traits >::gptr(), is_open(), and seekoff().

in_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ()  [inline], [inherited]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

is_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::is_open () const throw ( )   [inline], [nodiscard]
```

Returns true if the external file is open.
Referenced by __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), close(), imbue(), open(), seekoff(), seekpos(), setbuf(), and showmanyc().

open() [1/3]

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, __filebuf_type * > std::basic_filebuf< _CharT, _Traits >::open (
            const _Path & __s,
            ios_base::openmode __mode)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file, as a filesystem::path.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

open() [2/3]

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
            const char * __s,
            ios_base::openmode __mode)
```

Opens an external file.

Parameters

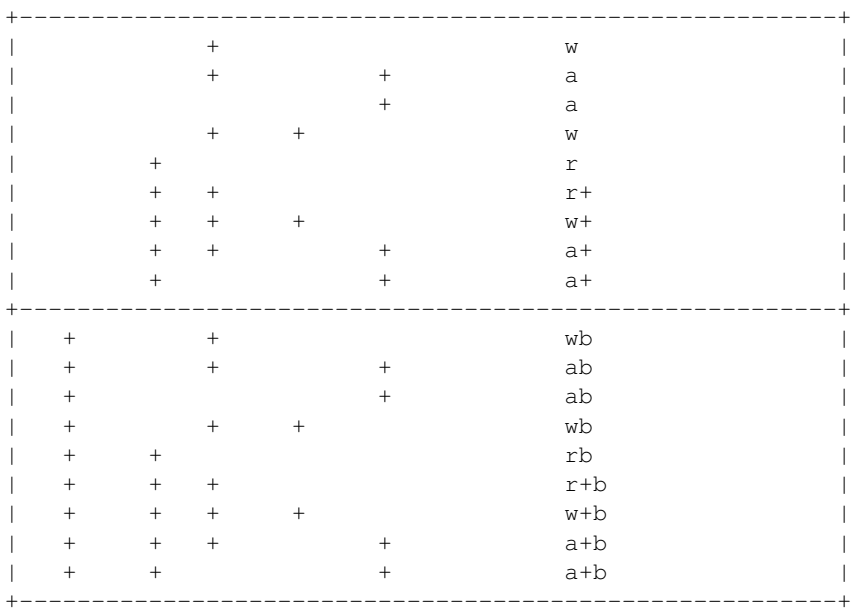
__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *__s* using the flags given in *__mode*.
Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596)

```
+-------------------------------------------------------+
| ios_base Flag combination            stdio equivalent   |
|binary  in  out  trunc  app                              |
```



References _M_mode, _M_reading, _M_set_buffer(), std::ios_base::ate, close(), std::ios_base::end, is_open(), and seekoff().
Referenced by std::basic_filebuf< char_type, traits_type >::open(), and std::basic_filebuf< char_type, traits_type >::open().

open() [3/3]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
            const std::string & __s,
            ios_base::openmode __mode)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

overflow()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::overflow (
            int_type __c = _Traits::eof())  [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

_↩ _c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually __c, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if __c is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_destroy_pback(), _M_mode, _M_reading, _M_set_buffer(), std::ios_base::app, std::ios_base::cur, std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase(), std::basic_streambuf< _CharT, _Traits >::pbum and std::basic_streambuf< _CharT, _Traits >::pptr().
Referenced by pbackfail(), sync(), underflow(), and xsgetn().

pbackfail()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::pbackfail (
            int_type __c = _Traits::eof())  [protected], [virtual]
```

Tries to back up the input sequence.

Parameters

_↩ _c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_create_pback(), _M_mode, _M_pback_init, _M_reading, _M_set_buffer(), std::ios_base::cur, std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::gbump(), std::basic_streambuf< _CharT, _ std::ios_base::in, overflow(), seekoff(), and underflow().

pbase()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, std::basic_stringbuf< char >::str(), std::basic_filebuf< _CharT, _Traits >::sync(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), and std::basic_streambuf< char_type, traits_type >::sputc().

pptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_streambuf< char_t std::basic_filebuf< _CharT, _Traits >::sync(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
            const locale & __loc)  [inline], [inherited]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ()  [inline], [inherited]
```

Calls virtual sync function.
Referenced by std::basic_istream< _CharT, _Traits >::sync().

sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_streambuf< char_type, traits_type >::snextc().

seekoff()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekoff (
            off_type ,
            ios_base::seekdir ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [protected], [virtual]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_destroy_pback(), _M_reading, std::ios_base::cur, is_open(), std::basic_streambuf< _CharT, _Traits >::pbase(), and std::basic_streambuf< _CharT, _Traits >::pptr().
Referenced by imbue(), open(), and pbackfail().

seekpos()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekpos (
            pos_type ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [protected], [virtual]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_destroy_pback(), std::ios_base::beg, and is_open().

setbuf()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__streambuf_type * std::basic_filebuf< _CharT, _Traits >::↩
setbuf (
            char_type * __s,
            streamsize __n)  [protected], [virtual]
```

Manipulates the buffer.

Parameters

_↩ _s	Pointer to a buffer area.
_↩ _n	Size of __s.

Returns

this

If no file has been opened, and both __s and __n are zero, then the stream becomes unbuffered. Otherwise, __s is used as a buffer; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html↩ #io.streambuf.buffering for more.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf, _M_buf_size, and is_open().

setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
            char_type * __gbeg,
            char_type * __gnext,
            char_type * __gend)  [inline], [protected], [inherited]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == eback(), __gnext == gptr(), and __gend == egptr()

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), std::basic_filebuf< char_type, traits_type >::_M_create std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
```

```
            char_type * __pbeg,
            char_type * __pend)  [inline], [protected], [inherited]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), and std::basic_filebuf< char_type, traits_type >::_M_s

sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::igr and std::basic_streambuf< char_type, traits_type >::snextc().

sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for xsgetn.

Parameters

_↩ _s	A buffer area.
_↩ _n	A count.

Returns xsgetn(__s,__n). The effect is to fill *__s*[0] through *__s*[__n-1] with characters from the input sequence, if possible.

showmanyc()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::showmanyc ()  [protected], [virtual]
```

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::ios_base::binary, std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits > std::ios_base::in, and is_open().

snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, char_traits< _CharT > >::get(), and std::basic_istream< _CharT, char_traits< _CharT > >::readsome().

sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
            char_type __c)  [inline], [inherited]
```

Pushing characters back into the input stream.

Parameters

_↩ _c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but __c is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be __c.
Referenced by std::basic_istream< _CharT, _Traits >::putback().

sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
            char_type __c)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _c	A character to output.

Returns

__c, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.
Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow().

sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
            const char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _s	A buffer read area.
_↩ _n	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __s[0] through __s[__n-1] to the output sequence, if possible.

sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ()  [inline], [inherited]
```

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.
Referenced by std::basic_istream< _CharT, _Traits >::unget().

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_filebuf< _CharT, _Traits >::sync ()  [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References overflow(), std::basic_streambuf< _CharT, _Traits >::pbase(), and std::basic_streambuf< _CharT, _Traits >::pptr().

uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ()  [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.
Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.
Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc(), and xsgetn().

underflow()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ()
[protected], [virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_destroy_pback(), _M_mode, _M_set_buffer(), std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::gptr(), std::ios_base::in, and overflow().
Referenced by pbackfail().

xsgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn (
            char_type * __s,
            streamsize __n)  [protected], [virtual]
```

Multiple character extraction.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __s[0] through __s[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __n characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_destroy_pback(), _M_mode, _M_pback_init, _M_reading, _M_set_buffer(), std::basic_streambuf< _CharT, _
std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::gbump(), std::basic_streambuf< _CharT, _T
std::ios_base::in, overflow(), std::basic_streambuf< _CharT, _Traits >::setg(), and std::basic_streambuf< char_type, traits_type >::xsgetn

xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual]
```

Multiple character insertion.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __s[0] through __s[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_mode, _M_reading, _M_set_buffer(), std::ios_base::app, std::basic_streambuf< _CharT, _Traits >::epptr(),
std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase(), std::basic_streambuf< _CharT, _Traits >::pptr(),
and std::basic_streambuf< char_type, traits_type >::xsputn().

6.233.4 Member Data Documentation

_M_buf

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_buf  [protected]
```

Pointer to the beginning of internal buffer.
Referenced by basic_filebuf(), and setbuf().

_M_buf_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale  [protected], [inherited]
```

Current locale setting.
Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

_M_buf_size

```
template<typename _CharT, typename _Traits>
size_t std::basic_filebuf< _CharT, _Traits >::_M_buf_size  [protected]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.
Referenced by basic_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::s overflow(), setbuf(), underflow(), xsgetn(), and xsputn().

_M_ext_buf

```
template<typename _CharT, typename _Traits>
char* std::basic_filebuf< _CharT, _Traits >::_M_ext_buf  [protected]
```

Buffer for external characters. Used for input when codecvt::always_noconv() == false. When valid, this corresponds to eback().
Referenced by basic_filebuf(), and imbue().

_M_ext_buf_size

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size  [protected]
```

Size of buffer held by _M_ext_buf.
Referenced by basic_filebuf().

_M_ext_next

```
template<typename _CharT, typename _Traits>
const char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next  [protected]
```

Pointers into the buffer held by _M_ext_buf that delimit a subsequence of bytes that have been read but not yet converted. When valid, _M_ext_next corresponds to egptr().
Referenced by basic_filebuf(), and imbue().

_M_in_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg  [protected], [inherited]
```

Start of get area.

_M_in_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur  [protected], [inherited]
```
Current read area.

_M_in_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end  [protected], [inherited]
```
End of get area.

_M_mode

```
template<typename _CharT, typename _Traits>
ios_base::openmode std::basic_filebuf< _CharT, _Traits >::_M_mode  [protected]
```
Place to stash in || out || in | out settings for current filebuf.
Referenced by basic_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::
close(), imbue(), open(), overflow(), pbackfail(), showmanyc(), underflow(), xsgetn(), and xsputn().

_M_out_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg  [protected], [inherited]
```
Start of put area.

_M_out_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur  [protected], [inherited]
```
Current put area.

_M_out_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end  [protected], [inherited]
```
End of put area.

_M_pback

```
template<typename _CharT, typename _Traits>
char_type std::basic_filebuf< _CharT, _Traits >::_M_pback  [protected]
```
Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf().

_M_pback_cur_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save  [protected]
```
Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf().

_M_pback_end_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save  [protected]
```

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf().

_M_pback_init

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_pback_init  [protected]
```

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf(), close(), pbackfail(), and xsgetn().

_M_reading

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_reading  [protected]
```

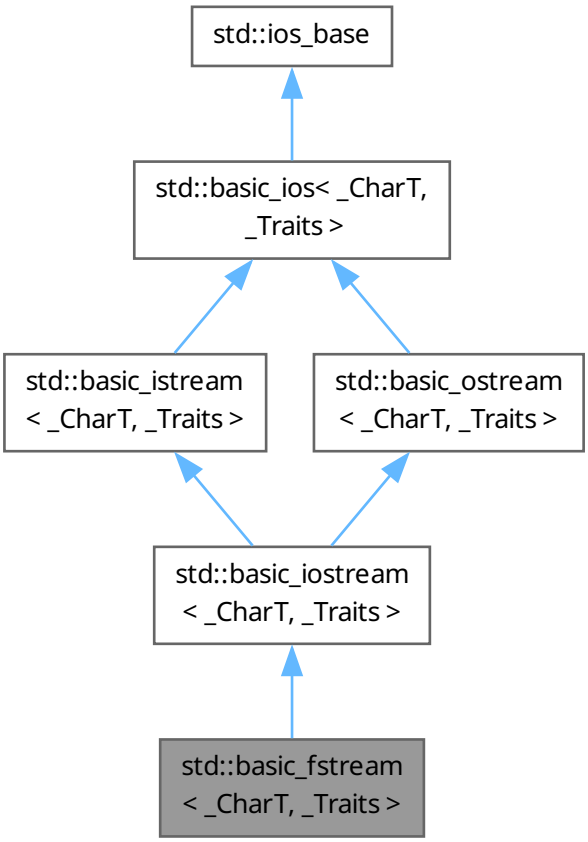
_M_reading == false && _M_writing == false for **uncommitted** mode; _M_reading == true for **read** mode; _M_writing == true for **write** mode;
NB: _M_reading == true && _M_writing == true is unused.
Referenced by basic_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::
close(), imbue(), open(), overflow(), pbackfail(), seekoff(), xsgetn(), and xsputn().
The documentation for this class was generated from the following files:

- fstream
- fstream.tcc

6.234 std::basic_fstream< _CharT, _Traits > Class Template Reference

```
#include <fstream>
```

Inheritance diagram for std::basic_fstream< _CharT, _Traits >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef ctype< _CharT > **__ctype_type**
- typedef basic_filebuf< char_type, traits_type > **__filebuf_type**
- typedef basic_ios< char_type, traits_type > **__ios_type**
- typedef basic_iostream< char_type, traits_type > **__iostream_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< _CharT, _Traits > **__ostream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags

- typedef traits_type::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

Public Member Functions

- basic_fstream ()
- **basic_fstream** (basic_fstream &&__rhs)
- template<typename _Path, typename _Require = _If_fs_path<_Path>>
  basic_fstream (const _Path &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- **basic_fstream** (const basic_fstream &)=delete
- basic_fstream (const char *__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- basic_fstream (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- ~basic_fstream ()
- template<typename _ValueT>
  basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- const locale & _M_getloc () const
- template<typename _ValueT>
  basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- void close ()
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- streamsize gcount () const
- basic_istream< wchar_t > & **getline** (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)
- locale getloc () const
- bool good () const
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- locale imbue (const locale &__loc)
- bool is_open ()
- bool **is_open** () const
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const

- template<typename _Path>
  _If_fs_path< _Path, void > open (const _Path &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- void open (const char *__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- void open (const std::string &__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- __ostream_type & operator<< (__streambuf_type *__sb)
- __ostream_type & operator<< (const void *__p)
- __ostream_type & **operator**<< (const volatile void *__p)
- __ostream_type & **operator**<< (nullptr_t)
- basic_fstream & **operator=** (basic_fstream &&__rhs)
- basic_fstream & **operator=** (const basic_fstream &)=delete
- __istream_type & operator>> (__streambuf_type *__sb)
- __istream_type & operator>> (void *&__p)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- __filebuf_type * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- __ostream_type & seekp (pos_type)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- void **swap** (basic_fstream &__rhs)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(*__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)

- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type *__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type *__s, streamsize __n)
- __istream_type & ignore (streamsize __n, int_type __delim)
- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- __istream_type & ignore (streamsize __n, char __delim)
- int_type peek ()
- __istream_type & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()
- __istream_type & seekg (pos_type)
- __istream_type & seekg (off_type, ios_base::seekdir)

- operator bool () const
- bool operator! () const

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- __ostream_type & write (const char_type *__s, streamsize __n)

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left

- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT>
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT>
  __ostream_type & **_M_insert** (_ValueT __v)
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_iostream &__rhs)
- void **swap** (basic_istream &__rhs)
- void **swap** (basic_ostream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**

- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

6.234.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_fstream< _CharT, _Traits >

Controlling input and output for files.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class supports reading from and writing to named files, using the inherited functions from std::basic_iostream. To control the associated sequence, an instance of std::basic_filebuf is used, which this page refers to as `sb`.

6.234.2 Member Typedef Documentation

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_e	One of the members of the event enum.
_b	Reference to the ios_base object.
_i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags  [inherited]
```

This is a bitmask type.
_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha

- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

iostate

`typedef _Ios_Iostate std::ios_base::iostate [inherited]`
This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

`typedef _Ios_Openmode std::ios_base::openmode [inherited]`
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir  [inherited]
```

This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

6.234.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

6.234.4 Constructor & Destructor Documentation

basic_fstream() [1/4]

```
template<typename _CharT, typename _Traits>
std::basic_fstream< _CharT, _Traits >::basic_fstream ()  [inline]
```

Default constructor.
Initializes sb using its default constructor, and passes &sb to the base class initializer. Does not open any files (you haven't given it a filename to open).

basic_fstream() [2/4]

```
template<typename _CharT, typename _Traits>
std::basic_fstream< _CharT, _Traits >::basic_fstream (
            const char * __s,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [explicit]
```

Create an input/output file stream.

Parameters

__s	Null terminated string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

basic_fstream() [3/4]

```
template<typename _CharT, typename _Traits>
std::basic_fstream< _CharT, _Traits >::basic_fstream (
            const std::string & __s,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [explicit]
```

Create an input/output file stream.

Parameters

__s	Null terminated string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

basic_fstream() [4/4]

```
template<typename _CharT, typename _Traits>
template<typename _Path, typename _Require = _If_fs_path<_Path>>
std::basic_fstream< _CharT, _Traits >::basic_fstream (
            const _Path & __s,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline]
```

Create an input/output file stream.

Parameters

__s	filesystem::path specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

~basic_fstream()

```
template<typename _CharT, typename _Traits>
std::basic_fstream< _CharT, _Traits >::~basic_fstream ()  [inline]
```

The destructor does nothing.
The file is closed by the filebuf object, not the formatting stream.

6.234.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
             iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::basic_fstream< char >::open(), std::basic_fstream< char >::
std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

close()

```
template<typename _CharT, typename _Traits>
void std::basic_fstream< _CharT, _Traits >::close ()  [inline]
```

Close the file.
Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
             const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::o

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<, std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()  [inherited]
```

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.
References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits
Referenced by std::flush().

gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const  [inline], [inherited]
```

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

get() [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().
Referenced by std::basic_istream< char_type, traits_type >::get(), and std::basic_istream< char_type, traits_type >::get().

get() [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb)  [inline], [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

*this

Returns `get`(__sb,widen('\n')).

get() [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb,
            char_type __delim)  [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

*this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

get() [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type & __c)  [inherited]
```

Simple extraction.

Parameters

_↩ _c	The character in which to store data.

Returns

*this

Tries to extract a character and store it in __c. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Simple multiple-character extraction.

Parameters

_↩ _s	Pointer to an array.
_↩ _n	Maximum number of characters to store in *s*.

Returns

*this

Returns get(__s,__n,widen('\n')).

get() [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.

__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

*this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.
In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

getline() [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
             char_type * __s,
             streamsize __n)  [inline], [inherited]
```

String extraction.

Parameters

_↩ _s	A character array in which to store the data.
_↩ _n	Maximum number of characters to extract.

Returns

*this

Returns `getline`(__s,__n,widen('\n')).

getline() [2/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
             char_type * __s,
             streamsize __n,
             char_type __delim)  [inherited]
```

Explicit specialization declarations, defined in src/istream.cc.

getline() [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)
In any case, a null character is stored in the next location in the array.
References _M_gcount, and std::ios_base::goodbit.
Referenced by std::basic_istream< char_type, traits_type >::getline().

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

ignore() [1/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >::sbumpc().
Referenced by std::basic_istream< char_type, traits_type >::ignore().

ignore() [2/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n)  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

ignore() [3/4]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            char __delim)  [inline], [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

ignore() [4/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            int_type __delim)  [inherited]
```

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide four overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩
html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_ios< char_type, traits_type >::basic_ios(), std::basic_istream< char_type, traits_type >::basic_istream(),

std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ostream< char_type, traits_type >::basic_ostream(), std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringstream< char >::basic_stringstream(), std::basic_stringstream< char >::basic_stringstream(), and std::basic_stringstream< char >::basic_stringstream().

is_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_fstream< _CharT, _Traits >::is_open ()  [inline], [nodiscard]
```

Wrapper to test for an open file.

Returns

```
rdbuf()->is_open()
```

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

open() [1/3]

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, void > std::basic_fstream< _CharT, _Traits >::open (
            const _Path & __s,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file, as a filesystem::path.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode)`. If that function fails, `failbit` is set in the stream's error state.

open() [2/3]

```
template<typename _CharT, typename _Traits>
void std::basic_fstream< _CharT, _Traits >::open (
            const char * __s,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode)`. If that function fails, `failbit` is set in the stream's error state.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), and std::basic_fstream< char >::open().

open() [3/3]

```
template<typename _CharT, typename _Traits>
void std::basic_fstream< _CharT, _Traits >::open (
            const std::string & __s,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode)`. If that function fails, `failbit` is set in the stream's error state.

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator<<() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ostream_type &(* __pf )(__ostream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            __streambuf_type * __sb)  [inherited]
```

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If __sb is NULL, the stream will set failbit in its error state.
Characters are extracted from __sb and inserted into *this until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from __sb, which sets failbit in the error state

If the function inserts no characters, failbit is set.

operator<<() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            bool __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            const void * __p)  [inline], [inherited]
```

Pointer arithmetic inserters.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            float __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            int __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            short __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.
References basic_ostream(), std::ios_base::badbit, std::ostreambuf_iterator< _CharT, _Traits >::failed(), std::ios_base::goodbit, std::num_put< _CharT, _OutIter >::put(), std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator<<() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned int __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned short __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator>>() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws.
For more information, see the iomanip header.

operator>>() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __istream_type &(* __pf )(__istream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws.
For more information, see the iomanip header.

operator>>() [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            __streambuf_type * __sb)  [inherited]
```

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If __sb is NULL, the stream will set failbit in its error state.
Characters are extracted from this stream and inserted into the __sb streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.
References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            bool & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            float & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            int & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            short & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator>>() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned int & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned short & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            void *& __p)  [inline], [inherited]
```

Basic arithmetic extractors.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

peek()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
            void )  [inherited]
```

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, good() is false, returns traits::eof(). Otherwise reads but does not extract the next input character.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

put()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
            char_type __c)  [inherited]
```

Simple insertion.

Parameters

_↩ _c	The character to insert.

Returns

*this

Tries to insert __c.

Note

This function is not overloaded on signed char and unsigned char.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits
Referenced by std::endl(), and std::ends().

putback()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
            char_type __c)  [inherited]
```

Unextracting a single character.

Parameters

_↩ _c	The character to push back into the input stream.

Returns

*this

If rdbuf() is not null, calls rdbuf()->sputbackc(c).
If rdbuf() is null or if sputbackc() fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to gcount() will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().
Referenced by std::operator<<().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_fstream< _CharT, _Traits >::rdbuf () const  [inline], [nodiscard]
```

Accessing the underlying buffer.

Returns

The current basic_filebuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::| std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type > and std::basic_istream< _CharT, _Traits >::unget().

read()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction without delimiters.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount.

readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction until the buffer is exhausted, but no more.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into __s depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::
and std::basic_ios< _CharT, _Traits >::setstate().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            pos_type __pos)  [inherited]
```

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            pos_type __pos)  [inherited]
```

Changing the current write position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
             iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),
std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(),
std::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::ope
std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >
std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
             void )  [inherited]
```

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.
Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(),
std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

sync_with_stdio()

```
bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellg()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
            void )  [inherited]
```

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

tellp()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()  [inherited]
```

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.
References std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf()

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unget()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
            void )  [inherited]
```

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.
If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sungetc().
Referenced by std::__detail::operator>>().

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Referenced by std::endl(), std::basic_ios< char_type, traits_type >::fill(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::getline(), std::getline(), std::getline(), std::operator<<(), std::operator<<(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

write()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
            const char_type * __s,
            streamsize __n)  [inherited]
```

Character string insertion.

Parameters

_↩ _s	The array to insert.
_↩ _n	Maximum number of characters to insert.

Returns

*this

Characters are copied from __s and inserted into the stream until one of the following happens:

- __n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

xalloc()

```
int std::ios_base::xalloc () throw ( )   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

6.234.6 Member Data Documentation

_M_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount  [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see gcount().
Referenced by get(), get(), get(), get(), getline(), ignore(), ignore(), ignore(), putback(), read(), readsome(), and unget().

adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```
A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app [static], [inherited]
```
Seek to end before each write.
Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate [static], [inherited]
```
Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_typ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

```
const seekdir std::ios_base::beg [static], [inherited]
```
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

```
const openmode std::ios_base::binary [static], [inherited]
```
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

```
const fmtflags std::ios_base::boolalpha  [static], [inherited]
```

Insert/extract `bool` in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

```
const seekdir std::ios_base::cur  [static], [inherited]
```

Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

```
const fmtflags std::ios_base::dec  [static], [inherited]
```

Converts integer input or generates integer output in decimal base.
Referenced by std::dec(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::opera and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>.

end

```
const seekdir std::ios_base::end  [static], [inherited]
```

Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

```
const iostate std::ios_base::eofbit  [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >: std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Tra std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, char_traits std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit  [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char > std::basic_ofstream< char >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_iostream< char_type, traits_type >::seekg(),

std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```

Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```

Indicates all is well.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >:
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(),
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore()
std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_
std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(),
std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::
std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _T
std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, char_traits< _CharT > >::tellg(), std::basic_istream< _Cha
and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```

Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_istringstream< char >::basic_istringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char >
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ifstream< char >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pb
std::basic_streambuf< char_type, traits_type >::pubseekoff(), std::basic_streambuf< char_type, traits_type >::pubseekpos(),
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_streambuf< char_type, traits_
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_streambuf< char_type, trait
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< char >::seekpos(), std::basic_filebuf< _CharT, _Traits >
std::basic_stringbuf< char >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underfl
std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

`const fmtflags std::ios_base::internal [static], [inherited]`
Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

`const fmtflags std::ios_base::left [static], [inherited]`
Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::left(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x

oct

`const fmtflags std::ios_base::oct [static], [inherited]`
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

`const openmode std::ios_base::out [static], [inherited]`
Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofs
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::o
std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::pubseekoff(),
std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_streambuf< char_type, traits_type >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_streambuf< char_type, traits_type >::seekpos(), std::basic_stringbuf< _Cha
std::basic_stringbuf< char >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsput

right

`const fmtflags std::ios_base::right [static], [inherited]`
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).
Referenced by std::right().

scientific

`const fmtflags std::ios_base::scientific [static], [inherited]`
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

`const fmtflags std::ios_base::showbase [static], [inherited]`
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

`const fmtflags std::ios_base::showpoint [static], [inherited]`
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

`const fmtflags std::ios_base::showpos [static], [inherited]`
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

`const fmtflags std::ios_base::skipws [static], [inherited]`
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x
and std::skipws().

trunc

`const openmode std::ios_base::trunc [static], [inherited]`
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

`const fmtflags std::ios_base::unitbuf [static], [inherited]`
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

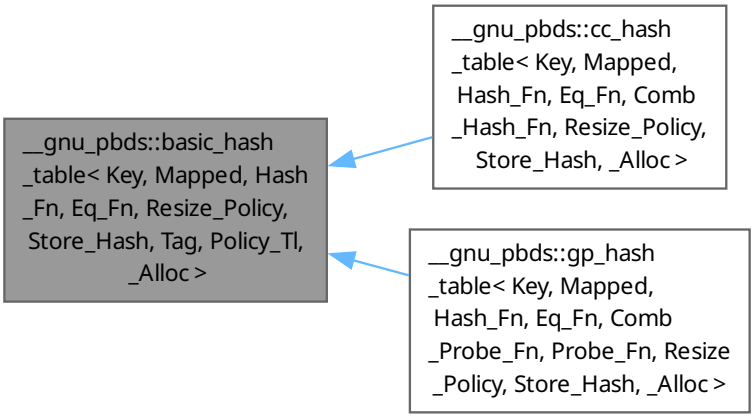
`const fmtflags std::ios_base::uppercase [static], [inherited]`
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following file:

- fstream

6.235 __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc > Class Template Reference

`#include <assoc_container.hpp>`
Inheritance diagram for __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash,

Tag, Policy_Tl, _Alloc >:



Protected Member Functions

- **basic_hash_table** (const basic_hash_table &other)
- template<typename T0>
  basic_hash_table (T0 t0)
- template<typename T0, typename T1>
  basic_hash_table (T0 t0, T1 t1)
- template<typename T0, typename T1, typename T2>
  basic_hash_table (T0 t0, T1 t1, T2 t2)
- template<typename T0, typename T1, typename T2, typename T3>
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3)
- template<typename T0, typename T1, typename T2, typename T3, typename T4>
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5>
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6>
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7>
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7)
- template<typename T0, typename T1, typename T2, typename T3, typename T4, typename T5, typename T6, typename T7, typename T8>
  basic_hash_table (T0 t0, T1 t1, T2 t2, T3 t3, T4 t4, T5 t5, T6 t6, T7 t7, T8 t8)

6.235.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename Resize_Policy, bool Store_Hash, typename Tag, typename Policy_Tl, typename _Alloc>
class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >

A hashed container abstraction.

Template Parameters

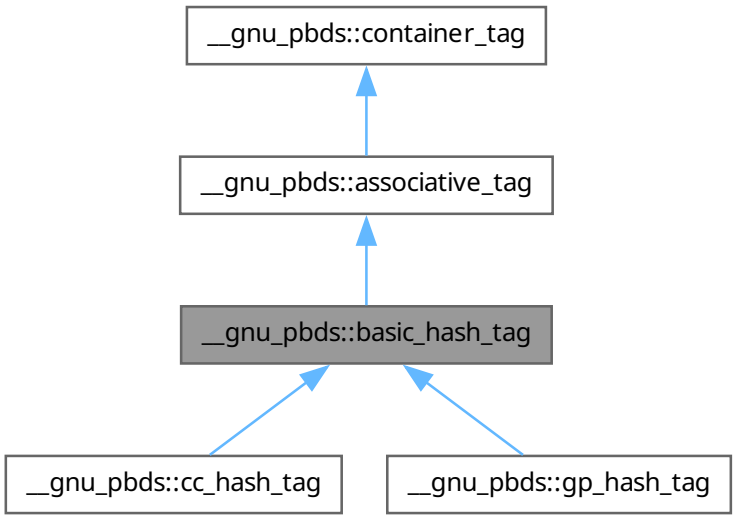
Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor.
Eq_Fn	Equal functor.
Resize_Policy	Resizes hash.
Store_Hash	Indicates whether the hash value will be stored along with each key.
Tag	Instantiating data structure type, see container_tag.
Policy_TL	Policy typelist.
_Alloc	Allocator type.

Base is dispatched at compile time via Tag, from the following choices: cc_hash_tag, gp_hash_tag, and descendants of basic_hash_tag.
Base choices are: detail::cc_ht_map, detail::gp_ht_map
The documentation for this class was generated from the following file:

- assoc_container.hpp

6.236 __gnu_pbds::basic_hash_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::basic_hash_tag:



6.236.1 Detailed Description

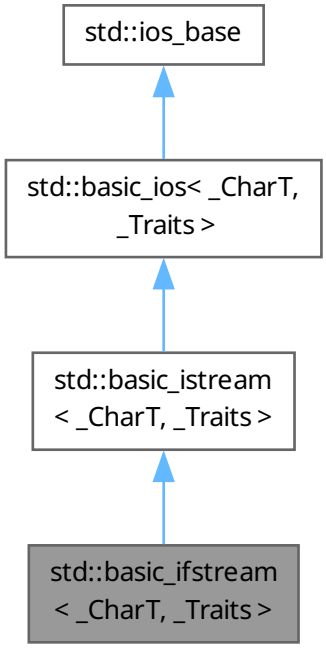
Basic hash structure.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.237 std::basic_ifstream< _CharT, _Traits > Class Template Reference

`#include <fstream>`

Inheritance diagram for std::basic_ifstream< _CharT, _Traits >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_filebuf< char_type, traits_type > **__filebuf_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_istream< char_type, traits_type > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef traits_type::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir

- typedef _Traits **traits_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type

Public Member Functions

- basic_ifstream ()
- **basic_ifstream** (basic_ifstream &&__rhs)
- template<typename _Path, typename _Require = _If_fs_path<_Path>>
  basic_ifstream (const _Path &__s, ios_base::openmode __mode=ios_base::in)
- **basic_ifstream** (const basic_ifstream &)=delete
- basic_ifstream (const char *__s, ios_base::openmode __mode=ios_base::in)
- basic_ifstream (const std::string &__s, ios_base::openmode __mode=ios_base::in)
- ~basic_ifstream ()
- template<typename _ValueT>
  basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- void close ()
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- streamsize gcount () const
- basic_istream< wchar_t > & **getline** (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)
- locale getloc () const
- bool good () const
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- locale imbue (const locale &__loc)
- bool is_open ()
- bool **is_open** () const
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- template<typename _Path>
  _If_fs_path< _Path, void > open (const _Path &__s, ios_base::openmode __mode=ios_base::in)
- void open (const char *__s, ios_base::openmode __mode=ios_base::in)
- void open (const std::string &__s, ios_base::openmode __mode=ios_base::in)
- basic_ifstream & **operator=** (basic_ifstream &&__rhs)
- basic_ifstream & **operator=** (const basic_ifstream &)=delete
- __istream_type & operator>> (__streambuf_type *__sb)

- __istream_type & operator>> (void *&__p)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- __filebuf_type * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- void **swap** (basic_ifstream &__rhs)
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(*__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type *__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type *__s, streamsize __n)
- __istream_type & ignore (streamsize __n, int_type __delim)
- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- __istream_type & ignore (streamsize __n, char __delim)
- int_type peek ()
- __istream_type & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()
- __istream_type & seekg (pos_type)
- __istream_type & seekg (off_type, ios_base::seekdir)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct

- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT>
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_istream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

6.237.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_ifstream< _CharT, _Traits >

Controlling input for files.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class supports reading from named files, using the inherited functions from std::basic_istream. To control the associated sequence, an instance of std::basic_filebuf is used, which this page refers to as `sb`.

6.237.2 Member Typedef Documentation

__num_put_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_put_type  [inherited]
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_e	One of the members of the event enum.
_b	Reference to the ios_base object.
_i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags  [inherited]
```

This is a bitmask type.
_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec

- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

iostate

`typedef _Ios_Iostate std::ios_base::iostate [inherited]`
This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

`typedef _Ios_Openmode std::ios_base::openmode [inherited]`
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir  [inherited]
```

This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

6.237.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

6.237.4 Constructor & Destructor Documentation

basic_ifstream() [1/4]

```
template<typename _CharT, typename _Traits>
std::basic_ifstream< _CharT, _Traits >::basic_ifstream ()  [inline]
```

Default constructor.
Initializes sb using its default constructor, and passes &sb to the base class initializer. Does not open any files (you haven't given it a filename to open).

basic_ifstream() [2/4]

```
template<typename _CharT, typename _Traits>
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
            const char * __s,
            ios_base::openmode __mode = ios_base::in)  [inline], [explicit]
```

Create an input file stream.

Parameters

__s	Null terminated string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

ios_base::in is automatically included in *__mode*.

basic_ifstream() [3/4]

```
template<typename _CharT, typename _Traits>
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
            const std::string & __s,
            ios_base::openmode __mode = ios_base::in)  [inline], [explicit]
```

Create an input file stream.

Parameters

__s	std::string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::in` is automatically included in *__mode*.

basic_ifstream() [4/4]

```
template<typename _CharT, typename _Traits>
template<typename _Path, typename _Require = _If_fs_path<_Path>>
std::basic_ifstream< _CharT, _Traits >::basic_ifstream (
            const _Path & __s,
            ios_base::openmode __mode = ios_base::in)  [inline]
```

Create an input file stream.

Parameters

__s	filesystem::path specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::in` is automatically included in *__mode*.

~basic_ifstream()

```
template<typename _CharT, typename _Traits>
std::basic_ifstream< _CharT, _Traits >::~basic_ifstream ()  [inline]
```

The destructor does nothing.
The file is closed by the filebuf object, not the formatting stream.

6.237.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
             iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::basic_fstream< char >::open(), std::basic_fstream< char >::
std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

close()

```
template<typename _CharT, typename _Traits>
void std::basic_ifstream< _CharT, _Traits >::close ()  [inline]
```

Close the file.
Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
             const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::o

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<, std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const  [inline], [inherited]
```

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

get() [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().
Referenced by std::basic_istream< char_type, traits_type >::get(), and std::basic_istream< char_type, traits_type >::get().

get() [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb)  [inline], [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

*this

Returns get(__sb,widen('\n')).

get() [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb,
            char_type __delim)  [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

*this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

get() [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type & __c)  [inherited]
```

Simple extraction.

Parameters

__c	The character in which to store data.

Returns

*this

Tries to extract a character and store it in *__c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Simple multiple-character extraction.

Parameters

_↩ _s	Pointer to an array.
_↩ _n	Maximum number of characters to store in *s*.

Returns

∗this

Returns `get`(__s,__n,widen('\n')).

get() [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in __s.
__delim	A "stop" character.

Returns

∗this

Characters are extracted and stored into __s until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals __*delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.
In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

getline() [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

String extraction.

Parameters

_↩ _s	A character array in which to store the data.
_↩ _n	Maximum number of characters to extract.

Returns

*this

Returns `getline`(__s,__n,widen('\n')).

getline() [2/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Explicit specialization declarations, defined in src/istream.cc.

getline() [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)
In any case, a null character is stored in the next location in the array.
References _M_gcount, and std::ios_base::goodbit.
Referenced by std::basic_istream< char_type, traits_type >::getline().

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```
Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```
Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

ignore() [1/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            void )  [inherited]
```
Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >::sbumpc().
Referenced by std::basic_istream< char_type, traits_type >::ignore().

ignore() [2/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n)  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

ignore() [3/4]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            char __delim)  [inline], [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

ignore() [4/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            int_type __delim)  [inherited]
```

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide four overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_ios< char_type, traits_type >::basic_ios(), std::basic_istream< char_type, traits_type >::basic_istream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ostream< char_type, traits_type >::basic_ostream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringstream< char >::basic_stringstream(),
std::basic_stringstream< char >::basic_stringstream(), and std::basic_stringstream< char >::basic_stringstream().

is_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_ifstream< _CharT, _Traits >::is_open ()  [inline], [nodiscard]
```

Wrapper to test for an open file.

Returns

`rdbuf()->is_open()`

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html

open() [1/3]

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, void > std::basic_ifstream< _CharT, _Traits >::open (
            const _Path & __s,
            ios_base::openmode __mode = ios_base::in)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file, as a filesystem::path.

__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

open() [2/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ifstream< _CharT, _Traits >::open (
            const char * __s,
            ios_base::openmode __mode = ios_base::in)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.
Referenced by std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), and std::basic_ifstream< char >::open().

open() [3/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ifstream< _CharT, _Traits >::open (
            const std::string & __s,
            ios_base::openmode __mode = ios_base::in)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode|in)`. If that function fails, `failbit` is set in the stream's error state.

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
```

bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator>>() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __ios_type &(* __pf )(__ios_type &)) [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.
For more information, see the iomanip header.

operator>>() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __istream_type &(* __pf )(__istream_type &)) [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.
For more information, see the iomanip header.

operator>>() [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            __streambuf_type * __sb) [inherited]
```

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.
References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            bool & __n) [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            float & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            int & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.

Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            short & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator>>() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned int & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned short & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            void *& __p)  [inline], [inherited]
```

Basic arithmetic extractors.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

peek()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
            void )  [inherited]
```

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, good() is false, returns traits::eof(). Otherwise reads but does not extract the next input character.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

putback()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
            char_type __c)  [inherited]
```

Unextracting a single character.

Parameters

_↩ _c	The character to push back into the input stream.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.
If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().
Referenced by std::operator<<().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _*ix*	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_ifstream< _CharT, _Traits >::rdbuf () const  [inline], [nodiscard]
```

Accessing the underlying buffer.

Returns

The current basic_filebuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__*sb*	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::p std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type > and std::basic_istream< _CharT, _Traits >::unget().

read()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction without delimiters.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount.

readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction until the buffer is exhausted, but no more.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into __*s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::
and std::basic_ios< _CharT, _Traits >::setstate().

register_callback()

```
void std::ios_base::register_callback (
             event_callback __fn,
             int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
             off_type __off,
             ios_base::seekdir __dir)  [inherited]
```

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits and std::basic_ios< _CharT, _Traits >::setstate().

seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            pos_type __pos)  [inherited]
```

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
             iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),
std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(),
std::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::ope
std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >
std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
             void )  [inherited]
```

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.
Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

sync_with_stdio()

```
bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellg()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
            void )  [inherited]
```

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unget()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
            void )  [inherited]
```

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.
If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sungetc().
Referenced by std::__detail::operator>>().

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_↩ _c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).widen(c)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html
Referenced by std::endl(), std::basic_ios< char_type, traits_type >::fill(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::getline(), std::getline(), std::getline(), std::operator<<(), std::operator<<(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F and std::tr2::operator>>().

width() [1/2]

`streamsize std::ios_base::width () const [inline], [nodiscard], [inherited]`
Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide) [inline], [inherited]
```
Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

xalloc()

`int std::ios_base::xalloc () throw ( ) [static], [inherited]`
Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

6.237.6 Member Data Documentation

_M_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount  [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see gcount().
Referenced by get(), get(), get(), get(), getline(), ignore(), ignore(), ignore(), putback(), read(), readsome(), and unget().

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```

Seek to end before each write.
Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate  [static], [inherited]
```

Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit  [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_typ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

basefield

```
const fmtflags std::ios_base::basefield  [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

const seekdir std::ios_base::beg [static], [inherited]
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

const openmode std::ios_base::binary [static], [inherited]
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static], [inherited]
Insert/extract bool in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

const seekdir std::ios_base::cur [static], [inherited]
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

const fmtflags std::ios_base::dec [static], [inherited]
Converts integer input or generates integer output in decimal base.
Referenced by std::dec(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::opera and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>.

end

const seekdir std::ios_base::end [static], [inherited]
Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

const iostate std::ios_base::eofbit [static], [inherited]
Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:: std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Tra std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, char_traits std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit  [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char > std::basic_ofstream< char >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```

Generate floating-point output in fixed-point notation.

Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```

Indicates all is well.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _T std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, char_traits< _CharT > >::tellg(), std::basic_istream< _Cha and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```

Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_istringstream< char >::basic_istringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char >
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ifstream< char >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::p
std::basic_streambuf< char_type, traits_type >::pubseekoff(), std::basic_streambuf< char_type, traits_type >::pubseekpos(),
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_streambuf< char_type, traits
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_streambuf< char_type, trait
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< char >::seekpos(), std::basic_filebuf< _CharT, _Traits >
std::basic_stringbuf< char >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underfl
std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::left(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x

oct

```
const fmtflags std::ios_base::oct  [static], [inherited]
```

Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static], [inherited]
```

Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofs
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::o
std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::pubseekoff(),
std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_streambuf< char_type, traits_type >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_streambuf< char_type, traits_type >::seekpos(), std::basic_stringbuf< _Cha
std::basic_stringbuf< char >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsput

right

```
const fmtflags std::ios_base::right  [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).
Referenced by std::right().

scientific

const fmtflags std::ios_base::scientific [static], [inherited]
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

const fmtflags std::ios_base::showbase [static], [inherited]
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

const fmtflags std::ios_base::showpoint [static], [inherited]
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

const fmtflags std::ios_base::showpos [static], [inherited]
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

const fmtflags std::ios_base::skipws [static], [inherited]
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x
and std::skipws().

trunc

const openmode std::ios_base::trunc [static], [inherited]
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

const fmtflags std::ios_base::unitbuf [static], [inherited]
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

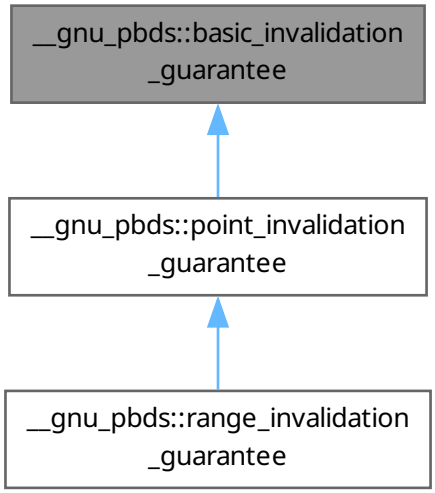
const fmtflags std::ios_base::uppercase [static], [inherited]
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following file:

- fstream

6.238 __gnu_pbds::basic_invalidation_guarantee Struct Reference

`#include <tag_and_trait.hpp>`

Inheritance diagram for __gnu_pbds::basic_invalidation_guarantee:



6.238.1 Detailed Description

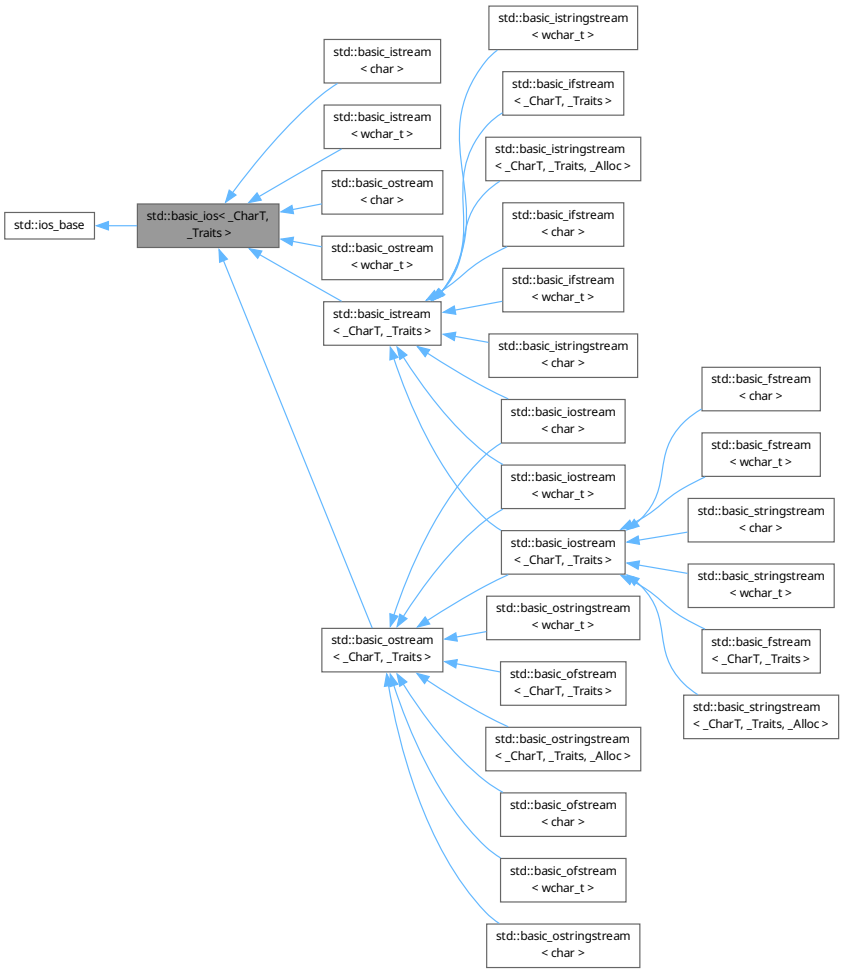
Signifies a basic invalidation guarantee that any iterator, pointer, or reference to a container object's mapped value type is valid as long as the container is not modified.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.239 std::basic_ios< _CharT, _Traits > Class Template Reference

`#include <basic_ios.h>`

Inheritance diagram for std::basic_ios< _CharT, _Traits >:



Public Types

- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(∗ event_callback) (event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef _Ios_Iostate iostate
- typedef _Ios_Openmode openmode
- typedef _Ios_Seekdir seekdir

- typedef _CharT char_type
- typedef _Traits::int_type int_type
- typedef _Traits::pos_type pos_type
- typedef _Traits::off_type off_type
- typedef _Traits traits_type

- typedef ctype< _CharT > __ctype_type
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type

Public Member Functions

- basic_ios (basic_streambuf< _CharT, _Traits > *__sb)
- virtual ~basic_ios ()
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- basic_ios ()
- **basic_ios** (const basic_ios &)=delete
- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)

- void **move** (basic_ios &__rhs)
- basic_ios & **operator=** (const basic_ios &)=delete
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

6.239.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_ios< _CharT, _Traits >

Template class basic_ios, virtual base class for all stream classes.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

Most of the member functions called dispatched on stream objects (e.g., `std::cout.foo(bar);`) are consolidated in this class.

6.239.2 Member Typedef Documentation

__ctype_type

```
template<typename _CharT, typename _Traits>
typedef ctype<_CharT> std::basic_ios< _CharT, _Traits >::__ctype_type
```

These are non-standard types.

__num_get_type

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_get_type
```

These are non-standard types.

__num_put_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_put_type
```

These are non-standard types.

char_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_ios< _CharT, _Traits >::char_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_↩ _*e*	One of the members of the event enum.
_↩ _*b*	Reference to the ios_base object.
_↩ _*i*	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags  [inherited]
```

This is a bitmask type.

_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right

- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

int_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::int_type std::basic_ios< _CharT, _Traits >::int_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

iostate

```
typedef _Ios_Iostate std::ios_base::iostate  [inherited]
```

This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

off_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::off_type std::basic_ios< _CharT, _Traits >::off_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

openmode

```
typedef _Ios_Openmode std::ios_base::openmode  [inherited]
```

This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate

- binary
- in
- out
- trunc

pos_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::pos_type std::basic_ios< _CharT, _Traits >::pos_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir  [inherited]
```

This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

traits_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_ios< _CharT, _Traits >::traits_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

6.239.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

6.239.4 Constructor & Destructor Documentation

basic_ios() [1/2]

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::basic_ios (
            basic_streambuf< _CharT, _Traits > * __sb)  [inline], [explicit]
```

Constructor performs initialization.
The parameter is passed by derived streams.
Referenced by copyfmt().

~basic_ios()

```
template<typename _CharT, typename _Traits>
virtual std::basic_ios< _CharT, _Traits >::~basic_ios ()  [inline], [virtual]
```

Empty.
The destructor does nothing. More specifically, it does not destroy the streambuf held by rdbuf().

basic_ios() [2/2]

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::basic_ios ()  [inline], [protected]
```

Empty.
The default constructor does nothing and is not normally accessible to users.

6.239.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
             iostate __state = goodbit)
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().

Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::basic_fstream< char >::open(), std::basic_fstream< char >::
std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios< _CharT, _Traits > & __rhs)
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >:: std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::o

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<,
std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::↩locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_ios< char_type, traits_type >::basic_ios(), std::basic_istream< char_type, traits_type >::basic_istream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ostream< char_type, traits_type >::basic_ostream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringstream< char >::basic_stringstream(),
std::basic_stringstream< char >::basic_stringstream(), and std::basic_stringstream< char >::basic_stringstream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _*ix*	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```
Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```
Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```
Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const  [inline],
[nodiscard]
```
Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.
Referenced by clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), imbue(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits > std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tel std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type > and std::basic_istream< _CharT, _Traits >::unget().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is ios_base::↩ adjustfield.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
             iostate __state)  [inline]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),
std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(),
std::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::ope
std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >
std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

sync_with_stdio()

```
bool std::ios_base::sync_with_stdio (
             bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/↩
libstdc++/manual/fstreams.html#std.io.filestreams.binary

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline]
```

Widens characters.

Parameters

_↩ _c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).widen(c)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html
Referenced by std::endl(), std::basic_ios< char_type, traits_type >::fill(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::getline(), std::getline(), std::getline(), std::operator<<(), std::operator<<(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```
Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline], [inherited]
```
Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

xalloc()

```
int std::ios_base::xalloc () throw ( )  [static], [inherited]
```
Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

6.239.6 Member Data Documentation

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```
A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```
Seek to end before each write.
Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

const openmode std::ios_base::ate [static], [inherited]
Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

const iostate std::ios_base::badbit [static], [inherited]
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_typ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

basefield

const fmtflags std::ios_base::basefield [static], [inherited]
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

const seekdir std::ios_base::beg [static], [inherited]
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

const openmode std::ios_base::binary [static], [inherited]
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩ filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static], [inherited]
Insert/extract `bool` in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

const seekdir std::ios_base::cur [static], [inherited]
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

```
const fmtflags std::ios_base::dec  [static], [inherited]
```

Converts integer input or generates integer output in decimal base.
Referenced by std::dec(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::opera
and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>.

end

```
const seekdir std::ios_base::end  [static], [inherited]
```

Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

```
const iostate std::ios_base::eofbit  [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:
std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(),
std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, char_traits
std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit  [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type,
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(),
std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(),
std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_iostream< char_type, traits_type >::seekg(),
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```

Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```

Indicates all is well.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _T std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, char_traits< _CharT > >::tellg(), std::basic_istream< _Cha and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```

Open for input. Default for `ifstream` and fstream.

Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea std::basic_istringstream< char >::basic_istringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char > std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::p std::basic_streambuf< char_type, traits_type >::pubseekoff(), std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_streambuf< char_type, traits std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_streambuf< char_type, trait std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< char >::seekpos(), std::basic_filebuf< _CharT, _Traits > std::basic_stringbuf< char >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underfl std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::left(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x

oct

`const fmtflags std::ios_base::oct [static], [inherited]`
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

`const openmode std::ios_base::out [static], [inherited]`
Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofs
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::o
std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::pubseekoff(),
std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_streambuf< char_type, traits_type >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_streambuf< char_type, traits_type >::seekpos(), std::basic_stringbuf< _Cha
std::basic_stringbuf< char >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsput

right

`const fmtflags std::ios_base::right [static], [inherited]`
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).
Referenced by std::right().

scientific

`const fmtflags std::ios_base::scientific [static], [inherited]`
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

`const fmtflags std::ios_base::showbase [static], [inherited]`
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

`const fmtflags std::ios_base::showpoint [static], [inherited]`
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

`const fmtflags std::ios_base::showpos [static], [inherited]`
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

`const fmtflags std::ios_base::skipws [static], [inherited]`
Skips leading white space before certain input operations.

Referenced by std::noskipws(), std::__detail::operator>>(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x
and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static], [inherited]
```
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static], [inherited]
```
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

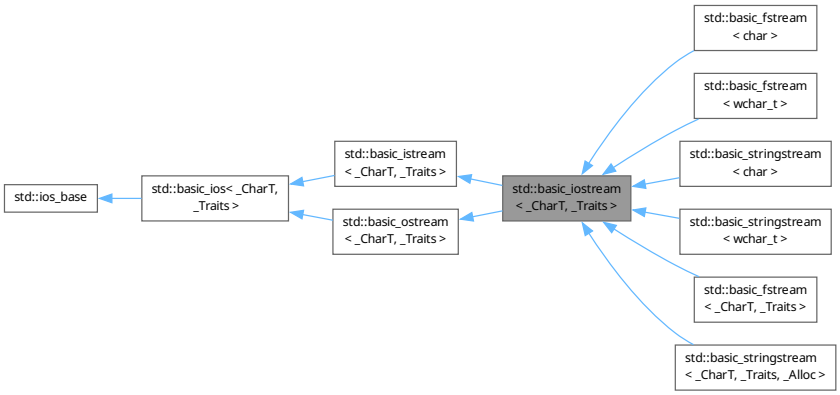
uppercase

```
const fmtflags std::ios_base::uppercase  [static], [inherited]
```
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following files:

- basic_ios.h
- basic_ios.tcc

6.240 std::basic_iostream< _CharT, _Traits > Class Template Reference

```
#include <istream>
```
Inheritance diagram for std::basic_iostream< _CharT, _Traits >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< _CharT, _Traits > **__ostream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef _Traits::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef _Traits::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef _Traits::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

Public Member Functions

- basic_iostream (basic_streambuf< _CharT, _Traits > *__sb)
- virtual ~basic_iostream ()
- template<typename _ValueT>
  basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- const locale & _M_getloc () const
- template<typename _ValueT>
  basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- streamsize gcount () const
- basic_istream< wchar_t > & **getline** (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)
- locale getloc () const
- bool good () const
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __ostream_type & operator<< (__streambuf_type *__sb)

- __ostream_type & operator<< (const void *__p)
- __ostream_type & **operator**<< (const volatile void *__p)
- __ostream_type & **operator**<< (nullptr_t)
- __istream_type & operator>> (__streambuf_type *__sb)
- __istream_type & operator>> (void *&__p)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- __ostream_type & seekp (pos_type)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(*__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type *__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type *__s, streamsize __n)
- __istream_type & ignore (streamsize __n, int_type __delim)
- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- __istream_type & ignore (streamsize __n, char __delim)
- int_type peek ()
- __istream_type & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()
- __istream_type & seekg (pos_type)
- __istream_type & seekg (off_type, ios_base::seekdir)

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- __ostream_type & write (const char_type *__s, streamsize __n)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint

- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- **basic_iostream** (basic_iostream &&__rhs)
- **basic_iostream** (const basic_iostream &)=delete
- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT>
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT>
  __ostream_type & **_M_insert** (_ValueT __v)
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- basic_iostream & **operator=** (basic_iostream &&__rhs)
- basic_iostream & **operator=** (const basic_iostream &)=delete
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_iostream &__rhs)
- void **swap** (basic_istream &__rhs)
- void **swap** (basic_ostream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**

- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

6.240.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_iostream< _CharT, _Traits >

Template class basic_iostream.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class multiply inherits from the input and output stream classes simply to provide a single interface.

6.240.2 Member Typedef Documentation

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```
The type of an event callback function.

Parameters

_e	One of the members of the event enum.
_b	Reference to the ios_base object.
_i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags  [inherited]
```
This is a bitmask type.
_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed

- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

iostate

```
typedef _Ios_Iostate std::ios_base::iostate  [inherited]
```

This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

```
typedef _Ios_Openmode std::ios_base::openmode  [inherited]
```

This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir  [inherited]
```

This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

6.240.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

6.240.4 Constructor & Destructor Documentation

basic_iostream()

```
template<typename _CharT, typename _Traits>
std::basic_iostream< _CharT, _Traits >::basic_iostream (
            basic_streambuf< _CharT, _Traits > * __sb)  [inline], [explicit]
```

Constructor does nothing.
Both of the parent classes are initialized with the same streambuf pointer passed to this constructor.

~basic_iostream()

```
template<typename _CharT, typename _Traits>
virtual std::basic_iostream< _CharT, _Traits >::~basic_iostream ()  [inline], [virtual]
```

Destructor does nothing.

6.240.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
             iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::basic_fstream< char >::open(), std::basic_fstream< char >::
std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
             const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::o

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<,
std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with __*fmtfl*.

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()  [inherited]
```

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.
References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits
Referenced by std::flush().

gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const  [inline], [inherited]
```

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

get() [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().
Referenced by std::basic_istream< char_type, traits_type >::get(), and std::basic_istream< char_type, traits_type >::get().

get() [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb)  [inline], [inherited]
```

Extraction into another streambuf.

Parameters

__*sb*	A streambuf in which to store data.

Returns

*this

Returns get(__sb,widen('\n')).

get() [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb,
            char_type __delim)  [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

*this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

get() [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type & __c)  [inherited]
```

Simple extraction.

Parameters

__c	The character in which to store data.

Returns

*this

Tries to extract a character and store it in *__c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Simple multiple-character extraction.

Parameters

_↩ _s	Pointer to an array.
_↩ _n	Maximum number of characters to store in *s*.

Returns

∗this

Returns `get(__s,__n,widen('\n'))`.

get() [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

∗this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.
In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

getline() [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
             char_type * __s,
             streamsize __n)  [inline], [inherited]
```

String extraction.

Parameters

_↩ _s	A character array in which to store the data.
_↩ _n	Maximum number of characters to extract.

Returns

*this

Returns getline(__s,__n,widen('\n')).

getline() [2/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
             char_type * __s,
             streamsize __n,
             char_type __delim)  [inherited]
```

Explicit specialization declarations, defined in src/istream.cc.

getline() [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
             char_type * __s,
             streamsize __n,
             char_type __delim)  [inherited]
```

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)
In any case, a null character is stored in the next location in the array.
References _M_gcount, and std::ios_base::goodbit.
Referenced by std::basic_istream< char_type, traits_type >::getline().

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

ignore() [1/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >::sbumpc().
Referenced by std::basic_istream< char_type, traits_type >::ignore().

ignore() [2/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n)  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

ignore() [3/4]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            char __delim)  [inline], [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

ignore() [4/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            int_type __delim)  [inherited]
```

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide four overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_ios< char_type, traits_type >::basic_ios(), std::basic_istream< char_type, traits_type >::basic_istream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ostream< char_type, traits_type >::basic_ostream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringstream< char >::basic_stringstream(),
std::basic_stringstream< char >::basic_stringstream(), and std::basic_stringstream< char >::basic_stringstream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator<<() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ostream_type &(* __pf )(__ostream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            __streambuf_type * __sb)  [inherited]
```

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

operator<<() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            bool __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            const void * __p)  [inline], [inherited]
```

Pointer arithmetic inserters.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            float __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            int __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.

Manipulators such as std::endl and std::hex use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            short __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.
References basic_ostream(), std::ios_base::badbit, std::ostreambuf_iterator< _CharT, _Traits >::failed(), std::ios_base::goodbit, std::num_put< _CharT, _OutIter >::put(), std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator<<() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned int __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned short __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator>>() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __istream_type &(* __pf )(__istream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            __streambuf_type * __sb)  [inherited]
```

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If __sb is NULL, the stream will set failbit in its error state.
Characters are extracted from this stream and inserted into the __sb streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or

- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.
References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            bool & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            float & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            int & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

operator>>() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            short & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator>>() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned int & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned short & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            void *& __p)  [inline], [inherited]
```

Basic arithmetic extractors.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

peek()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
            void )  [inherited]
```

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

put()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
            char_type __c)  [inherited]
```

Simple insertion.

Parameters

_↩ _c	The character to insert.

Returns

*this

Tries to insert __c.

Note

This function is not overloaded on signed char and unsigned char.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits
Referenced by std::endl(), and std::ends().

putback()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
            char_type __c)  [inherited]
```

Unextracting a single character.

Parameters

_↩ _c	The character to push back into the input stream.

Returns

∗this

If rdbuf() is not null, calls rdbuf()->sputbackc(c).
If rdbuf() is null or if sputbackc() fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to gcount() will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().
Referenced by std::operator<<().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const  [inline],
[nodiscard], [inherited]
```

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.
Referenced by clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), imbue(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits > std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tel std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type > and std::basic_istream< _CharT, _Traits >::unget().

read()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction without delimiters.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount.

readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction until the buffer is exhausted, but no more.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *__s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters

- if A > 0, extracts min(A,n)

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::
and std::basic_ios< _CharT, _Traits >::setstate().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If fail() is not true, calls rdbuf()->pubseekoff(__off,__dir). If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to gcount().

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            pos_type __pos)  [inherited]
```

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(),
and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            pos_type __pos)  [inherited]
```

Changing the current write position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),
std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(),
std::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::ope
std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >
std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
            void )  [inherited]
```

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.
Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

sync_with_stdio()

```
bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/↩
libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellg()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
            void )  [inherited]
```

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

tellp()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()  [inherited]
```

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.
References std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf()

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unget()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
            void )  [inherited]
```

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.
If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sungetc().
Referenced by std::__detail::operator>>().

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_↩ _c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
Referenced by std::endl(), std::basic_ios< char_type, traits_type >::fill(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::getline(), std::getline(), std::getline(), std::operator<<(), std::operator<<(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
             streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

write()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
             const char_type * __s,
             streamsize __n)  [inherited]
```

Character string insertion.

Parameters

_↩ _s	The array to insert.

_↩ _n	Maximum number of characters to insert.

Returns

*this

Characters are copied from __s and inserted into the stream until one of the following happens:

- __n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

xalloc()

```
int std::ios_base::xalloc () throw ( )   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

6.240.6 Member Data Documentation

_M_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount  [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see gcount().
Referenced by get(), get(), get(), get(), getline(), ignore(), ignore(), ignore(), putback(), read(), readsome(), and unget().

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of setf.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```

Seek to end before each write.
Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

const openmode std::ios_base::ate [static], [inherited]
Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

const iostate std::ios_base::badbit [static], [inherited]
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_typ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

basefield

const fmtflags std::ios_base::basefield [static], [inherited]
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

const seekdir std::ios_base::beg [static], [inherited]
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

const openmode std::ios_base::binary [static], [inherited]
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩ filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static], [inherited]
Insert/extract `bool` in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

const seekdir std::ios_base::cur [static], [inherited]
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.
Referenced by std::dec(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::opera
and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>.

end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:
std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(),
std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, char_traits
std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type,
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(),
std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(),
std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_iostream< char_type, traits_type >::seekg(),
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```

Indicates all is well.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _T std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, char_traits< _CharT > >::tellg(), std::basic_istream< _Cha and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```

Open for input. Default for `ifstream` and fstream.

Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea std::basic_istringstream< char >::basic_istringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char > std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::p std::basic_streambuf< char_type, traits_type >::pubseekoff(), std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_streambuf< char_type, traits std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_streambuf< char_type, trait std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< char >::seekpos(), std::basic_filebuf< _CharT, _Traits > std::basic_stringbuf< char >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underfl std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::left(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x

oct

```
const fmtflags std::ios_base::oct  [static], [inherited]
```
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static], [inherited]
```
Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofs
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::o
std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::pubseekoff(),
std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_streambuf< char_type, traits_type >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_streambuf< char_type, traits_type >::seekpos(), std::basic_stringbuf< _Cha
std::basic_stringbuf< char >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsput

right

```
const fmtflags std::ios_base::right  [static], [inherited]
```
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).
Referenced by std::right().

scientific

```
const fmtflags std::ios_base::scientific  [static], [inherited]
```
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

```
const fmtflags std::ios_base::showbase  [static], [inherited]
```
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

```
const fmtflags std::ios_base::showpoint  [static], [inherited]
```
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

```
const fmtflags std::ios_base::showpos  [static], [inherited]
```
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static], [inherited]
```
Skips leading white space before certain input operations.

Referenced by std::noskipws(), std::__detail::operator>>(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x
and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static], [inherited]
```
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static], [inherited]
```
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

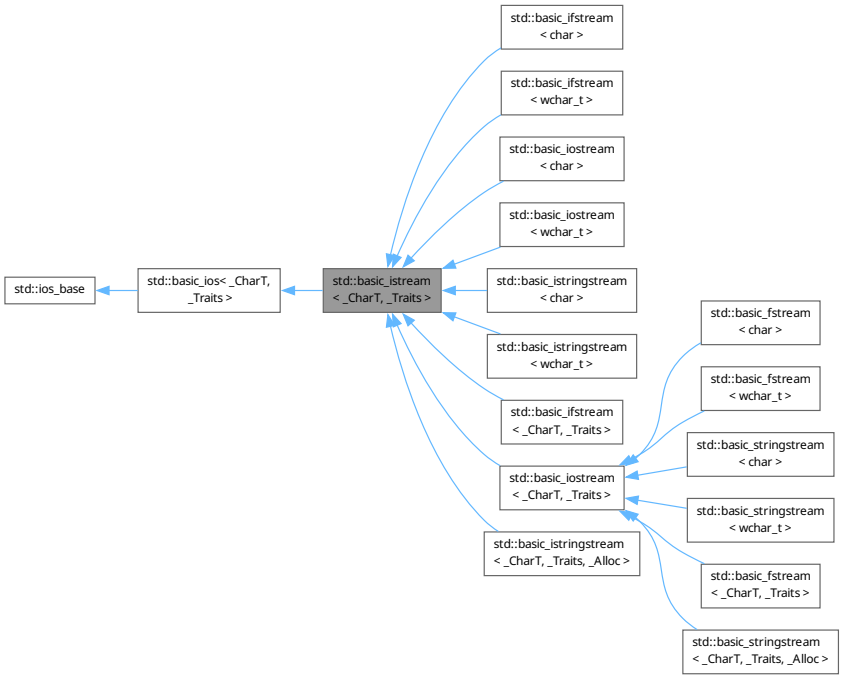
uppercase

```
const fmtflags std::ios_base::uppercase  [static], [inherited]
```
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following file:

- istream

6.241 std::basic_istream< _CharT, _Traits > Class Template Reference

`#include <istream>`
Inheritance diagram for std::basic_istream< _CharT, _Traits >:



Classes

- class sentry

Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef _Traits::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef _Traits::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef _Traits::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type

Public Member Functions

- basic_istream (__streambuf_type *__sb)
- virtual ~basic_istream ()
- template<typename _ValueT>
  basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- streamsize gcount () const
- basic_istream< wchar_t > & **getline** (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)
- locale getloc () const
- bool good () const
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)

- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __istream_type & operator>> (__streambuf_type *__sb)
- __istream_type & operator>> (void *&__p)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(*__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type *__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type *__s, streamsize __n)
- __istream_type & ignore (streamsize __n, int_type __delim)
- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- __istream_type & ignore (streamsize __n, char __delim)
- int_type peek ()
- __istream_type & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()
- __istream_type & seekg (pos_type)
- __istream_type & seekg (off_type, ios_base::seekdir)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end

- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- **basic_istream** (basic_istream &&__rhs)
- **basic_istream** (const basic_istream &)=delete
- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT>
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- basic_istream & **operator=** (basic_istream &&__rhs)
- basic_istream & **operator=** (const basic_istream &)=delete
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_istream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

Friends

- class **sentry**

6.241.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_istream< _CharT, _Traits >

Template class basic_istream.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This is the base class for all input streams. It provides text formatting of all builtin types, and communicates with any class derived from basic_streambuf to do the actual input.

6.241.2 Member Typedef Documentation

__num_put_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_put_type  [inherited]
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```
The type of an event callback function.

Parameters

_e	One of the members of the event enum.
_b	Reference to the ios_base object.
_i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags  [inherited]
```
This is a bitmask type.
_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

iostate

```
typedef _Ios_Iostate std::ios_base::iostate  [inherited]
```

This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

```
typedef _Ios_Openmode std::ios_base::openmode  [inherited]
```

This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir  [inherited]
```

This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

6.241.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

6.241.4 Constructor & Destructor Documentation

basic_istream()

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits >::basic_istream (
            __streambuf_type * __sb)  [inline], [explicit]
```

Base constructor.
This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry().

~basic_istream()

```
template<typename _CharT, typename _Traits>
virtual std::basic_istream< _CharT, _Traits >::~basic_istream ()  [inline], [virtual]
```

Base destructor.
This does very little apart from providing a virtual base dtor.

6.241.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
            iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::basic_fstream< char >::open(), std::basic_fstream< char >::
std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::o

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<,
std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const  [inline]
```

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

get() [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
            void )
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().
Referenced by std::basic_istream< char_type, traits_type >::get(), and std::basic_istream< char_type, traits_type >::get().

get() [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb)  [inline]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

*this

Returns get(__sb,widen('\n')).

get() [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb,
            char_type __delim)
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

*this

Characters are extracted and inserted into __*sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals __*delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

get() [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type & __c)
```

Simple extraction.

Parameters

_↩ _c	The character in which to store data.

Returns

*this

Tries to extract a character and store it in __*c*. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n)  [inline]
```

Simple multiple-character extraction.

Parameters

_↩ _s	Pointer to an array.
_↩ _n	Maximum number of characters to store in *s*.

Returns

*this

Returns `get(__s,__n,widen('\n'))`.

get() [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n,
            char_type __delim)
```

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

*this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.
In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

getline() [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n)  [inline]
```

String extraction.

Parameters

_↩ _s	A character array in which to store the data.
_↩ _n	Maximum number of characters to extract.

Returns

*this

Returns `getline(__s,__n,widen('\n'))`.

getline() [2/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)
```

Explicit specialization declarations, defined in src/istream.cc.

getline() [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)
```

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)
In any case, a null character is stored in the next location in the array.
References _M_gcount, and std::ios_base::goodbit.
Referenced by std::basic_istream< char_type, traits_type >::getline().

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```
Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```
Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

ignore() [1/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            void )
```
Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >::sbumpc().
Referenced by std::basic_istream< char_type, traits_type >::ignore().

ignore() [2/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n)
```
Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

ignore() [3/4]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            char __delim)  [inline]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

ignore() [4/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            int_type __delim)
```

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide four overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_ios< char_type, traits_type >::basic_ios(), std::basic_istream< char_type, traits_type >::basic_istream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ostream< char_type, traits_type >::basic_ostream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringstream< char >::basic_stringstream(),
std::basic_stringstream< char >::basic_stringstream(), and std::basic_stringstream< char >::basic_stringstream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator>>() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __ios_type &(* __pf )(__ios_type &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.
For more information, see the iomanip header.

operator>>() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __istream_type &(* __pf )(__istream_type &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.
For more information, see the iomanip header.

operator>>() [3/17]

```
template<typename _CharT, typename _Traits>
```

```
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            __streambuf_type * __sb)
```

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If __sb is NULL, the stream will set failbit in its error state.
Characters are extracted from this stream and inserted into the __sb streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.
References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            bool & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            double & __f)  [inline]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            float & __f)  [inline]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            int & __n)
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            ios_base &(* __pf )(ios_base &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`. For more information, see the iomanip header.

operator>>() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long double & __f)  [inline]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long long & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            short & __n)
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator>>() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned int & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long long & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned short & __n)  [inline]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            void *& __p)  [inline]
```

Basic arithmetic extractors.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

peek()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
            void )
```

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, good() is false, returns traits::eof(). Otherwise reads but does not extract the next input character.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

putback()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
            char_type __c)
```

Unextracting a single character.

Parameters

_↩ _c	The character to push back into the input stream.

Returns

∗this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.
If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().
Referenced by std::operator<<().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const  [inline],
[nodiscard], [inherited]
```

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.
Referenced by clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), imbue(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits > std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tel std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type > and std::basic_istream< _CharT, _Traits >::unget().

read()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
            char_type * __s,
            streamsize __n)
```

Extraction without delimiters.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount.

readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
            char_type * __s,
            streamsize __n)
```

Extraction until the buffer is exhausted, but no more.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into *__s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters

- if A > 0, extracts min(A,n)

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::
and std::basic_ios< _CharT, _Traits >::setstate().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            off_type __off,
            ios_base::seekdir __dir)
```

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If fail() is not true, calls rdbuf()->pubseekoff(__off,__dir). If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to gcount().

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(
std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits
and std::basic_ios< _CharT, _Traits >::setstate().

seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            pos_type __pos)
```

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),
std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(),
std::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::ope
std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >
std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
            void )
```

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.
Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(),
std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

sync_with_stdio()

```
bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellg()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
            void )
```

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unget()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
            void )
```

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.
If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sungetc().
Referenced by std::__detail::operator>>().

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Referenced by std::endl(), std::basic_ios< char_type, traits_type >::fill(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::getline(), std::getline(), std::getline(), std::operator<<(), std::operator<<(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
             streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

xalloc()

```
int std::ios_base::xalloc () throw ( )  [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

6.241.6 Member Data Documentation

_M_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount  [protected]
```

The number of characters extracted in the previous unformatted function; see gcount().
Referenced by get(), get(), get(), get(), getline(), ignore(), ignore(), ignore(), putback(), read(), readsome(), and unget().

adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```
A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app [static], [inherited]
```
Seek to end before each write.
Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate [static], [inherited]
```
Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_typ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

```
const seekdir std::ios_base::beg [static], [inherited]
```
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

```
const openmode std::ios_base::binary [static], [inherited]
```
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

```
const fmtflags std::ios_base::boolalpha  [static], [inherited]
```
Insert/extract `bool` in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

```
const seekdir std::ios_base::cur  [static], [inherited]
```
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

```
const fmtflags std::ios_base::dec  [static], [inherited]
```
Converts integer input or generates integer output in decimal base.
Referenced by std::dec(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::opera and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>.

end

```
const seekdir std::ios_base::end  [static], [inherited]
```
Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

```
const iostate std::ios_base::eofbit  [static], [inherited]
```
Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >: std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Tra std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, char_traits std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit  [static], [inherited]
```
Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char > std::basic_ofstream< char >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_iostream< char_type, traits_type >::seekg(),

std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```

Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```

Indicates all is well.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _T std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, char_traits< _CharT > >::tellg(), std::basic_istream< _Cha and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```

Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea std::basic_istringstream< char >::basic_istringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char > std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pb std::basic_streambuf< char_type, traits_type >::pubseekoff(), std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_streambuf< char_type, traits std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_streambuf< char_type, trait std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< char >::seekpos(), std::basic_filebuf< _CharT, _Traits > std::basic_stringbuf< char >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underfl std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

`const fmtflags std::ios_base::internal [static], [inherited]`
Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

`const fmtflags std::ios_base::left [static], [inherited]`
Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::left(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x

oct

`const fmtflags std::ios_base::oct [static], [inherited]`
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

`const openmode std::ios_base::out [static], [inherited]`
Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofs
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::o
std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::pubseekoff(),
std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_streambuf< char_type, traits_type >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_streambuf< char_type, traits_type >::seekpos(), std::basic_stringbuf< _Cha
std::basic_stringbuf< char >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsput

right

`const fmtflags std::ios_base::right [static], [inherited]`
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).
Referenced by std::right().

scientific

`const fmtflags std::ios_base::scientific [static], [inherited]`
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

`const fmtflags std::ios_base::showbase [static], [inherited]`
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

`const fmtflags std::ios_base::showpoint [static], [inherited]`
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

`const fmtflags std::ios_base::showpos [static], [inherited]`
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

`const fmtflags std::ios_base::skipws [static], [inherited]`
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x
and std::skipws().

trunc

`const openmode std::ios_base::trunc [static], [inherited]`
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

`const fmtflags std::ios_base::unitbuf [static], [inherited]`
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

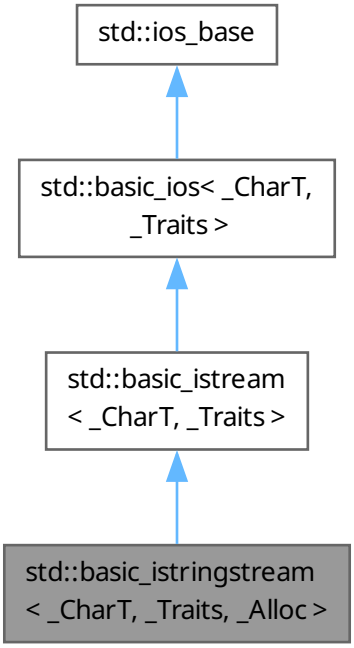
`const fmtflags std::ios_base::uppercase [static], [inherited]`
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following files:

- istream
- istream.tcc

6.242 std::basic_istringstream< _CharT, _Traits, _Alloc > Class Template Reference

`#include <sstream>`

Inheritance diagram for std::basic_istringstream< _CharT, _Traits, _Alloc >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_istream< char_type, traits_type > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef basic_string< _CharT, _Traits, _Alloc > **__string_type**
- typedef basic_stringbuf< _CharT, _Traits, _Alloc > **__stringbuf_type**
- typedef _Alloc **allocator_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef traits_type::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > __num_put_type

Public Member Functions

- basic_istringstream ()
- **basic_istringstream** (__string_type &&__str, ios_base::openmode __mode=ios_base::in)
- **basic_istringstream** (basic_istringstream &&__rhs)
- basic_istringstream (const __string_type &__str, ios_base::openmode __mode=ios_base::in)
- **basic_istringstream** (const basic_istringstream &)=delete
- template<typename _SAlloc>
  basic_istringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, const allocator_type &__a)
- template<typename _SAlloc>
  basic_istringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, ios_base::openmode __mode, const allocator_type &__a)
- template<typename _SAlloc>
  basic_istringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, ios_base::openmode __↩ mode=ios_base::in)
- basic_istringstream (ios_base::openmode __mode)
- **basic_istringstream** (ios_base::openmode __mode, const allocator_type &__a)
- ~basic_istringstream ()
- template<typename _ValueT>
  basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- const locale & _M_getloc () const
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- streamsize gcount () const
- basic_istream< wchar_t > & **getline** (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)
- locale getloc () const
- bool good () const
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- basic_istringstream & **operator=** (basic_istringstream &&__rhs)
- basic_istringstream & **operator=** (const basic_istringstream &)=delete
- __istream_type & operator>> (__streambuf_type *__sb)
- __istream_type & operator>> (void *&__p)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)

- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- __stringbuf_type * rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- __string_type **str** () &&
- __string_type str () const &
- void **str** (__string_type &&__s)
- void str (const __string_type &__s)
- template<__allocator_like _SAlloc>
  basic_string< _CharT, _Traits, _SAlloc > **str** (const _SAlloc &__sa) const
- template<__allocator_like _SAlloc>
  void **str** (const basic_string< _CharT, _Traits, _SAlloc > &__s)
- void **swap** (basic_istringstream &__rhs)
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- basic_string_view< char_type, traits_type > **view** () const noexcept
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))
- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(*__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type *__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type *__s, streamsize __n)
- __istream_type & ignore (streamsize __n, int_type __delim)
- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- __istream_type & ignore (streamsize __n, char __delim)
- int_type peek ()
- __istream_type & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()
- __istream_type & seekg (pos_type)
- __istream_type & seekg (off_type, ios_base::seekdir)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end

- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT>
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_istream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount

- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

6.242.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_istringstream< _CharT, _Traits, _Alloc >

Controlling input for std::string.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

This class supports reading from objects of type std::basic_string, using the inherited functions from std::basic_istream. To control the associated sequence, an instance of std::basic_stringbuf is used, which this page refers to as `sb`.

6.242.2 Member Typedef Documentation

__num_put_type

```
template<typename _CharT, typename _Traits>
typedef num_put<_CharT, ostreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_put_type  [inherited]
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_e	One of the members of the event enum.
_b	Reference to the ios_base object.

_↩ _*i*	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

fmtflags

`typedef _Ios_Fmtflags std::ios_base::fmtflags [inherited]`

This is a bitmask type.

_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

iostate

typedef _Ios_Iostate std::ios_base::iostate [inherited]
This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

typedef _Ios_Openmode std::ios_base::openmode [inherited]
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

typedef _Ios_Seekdir std::ios_base::seekdir [inherited]
This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

6.242.3 Member Enumeration Documentation

event

enum std::ios_base::event [inherited]
The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

6.242.4 Constructor & Destructor Documentation

basic_istringstream() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream ()  [inline]
```

Default constructor starts with an empty string buffer.
Initializes sb using in, and passes &sb to the base class initializer. Does not allocate any buffer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

basic_istringstream() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream (
            ios_base::openmode __mode)  [inline], [explicit]
```

Starts with an empty string buffer.

Parameters

__mode	Whether the buffer can read, or write, or both.

ios_base::in is automatically included in *__mode*.
Initializes sb using __mode|in, and passes &sb to the base class initializer. Does not allocate any buffer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

basic_istringstream() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istringstream< _CharT, _Traits, _Alloc >::basic_istringstream (
            const __string_type & __str,
            ios_base::openmode __mode = ios_base::in)  [inline], [explicit]
```

Starts with an existing string buffer.

Parameters

__str	A string to copy as a starting buffer.
__mode	Whether the buffer can read, or write, or both.

ios_base::in is automatically included in *mode*.
Initializes sb using *str* and mode|in, and passes &sb to the base class initializer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

~basic_istringstream()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_istringstream< _CharT, _Traits, _Alloc >::~basic_istringstream ()  [inline]
```

The destructor does nothing.
The buffer is deallocated by the stringbuf object, not the formatting stream.

6.242.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
             iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::basic_fstream< char >::open(), std::basic_fstream< char >::
std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
             const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
```

```
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >:
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::o

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
             char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<,
std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const  [inline], [inherited]
```

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

get() [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit,
std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().
Referenced by std::basic_istream< char_type, traits_type >::get(), and std::basic_istream< char_type, traits_type >::get().

get() [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb)  [inline], [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

*this

Returns `get`(__sb,widen('\n')).

get() [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb,
            char_type __delim)  [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

*this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

get() [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type & __c)  [inherited]
```

Simple extraction.

Parameters

__c	The character in which to store data.

Returns

*this

Tries to extract a character and store it in __c. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
             char_type * __s,
             streamsize __n)  [inline], [inherited]
```

Simple multiple-character extraction.

Parameters

_↩ _s	Pointer to an array.
_↩ _n	Maximum number of characters to store in *s*.

Returns

*this

Returns get(__s,__n,widen('\n')).

get() [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
             char_type * __s,
             streamsize __n,
             char_type __delim)  [inherited]
```

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in __s.
__delim	A "stop" character.

Returns

*this

Characters are extracted and stored into __*s* until one of the following happens:

- __n-1 characters are stored
- the input sequence reaches EOF
- the next character equals __*delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.
In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

getline() [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
             char_type * __s,
             streamsize __n)  [inline], [inherited]
```

String extraction.

Parameters

_↩ _*s*	A character array in which to store the data.
_↩ _*n*	Maximum number of characters to extract.

Returns

*this

Returns getline(__s,__n,widen('\n')).

getline() [2/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
             char_type * __s,
             streamsize __n,
             char_type __delim)  [inherited]
```

Explicit specialization declarations, defined in src/istream.cc.

getline() [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
             char_type * __s,
             streamsize __n,
             char_type __delim)  [inherited]
```

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)
In any case, a null character is stored in the next location in the array.
References _M_gcount, and std::ios_base::goodbit.
Referenced by std::basic_istream< char_type, traits_type >::getline().

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

ignore() [1/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >::sbumpc().
Referenced by std::basic_istream< char_type, traits_type >::ignore().

ignore() [2/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n)  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

ignore() [3/4]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            char __delim)  [inline], [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

ignore() [4/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            int_type __delim)  [inherited]
```

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if __*n* != `std::numeric_limits<int>::max()`, __*n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals __*delim* (in this case, the character is extracted); note that this condition will never occur if __*delim* equals `traits::eof()`.

NB: Provide four overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__*loc*	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩
html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_ios< char_type, traits_type >::basic_ios(), std::basic_istream< char_type, traits_type >::basic_istream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ostream< char_type, traits_type >::basic_ostream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringstream< char >::basic_stringstream(),
std::basic_stringstream< char >::basic_stringstream(), and std::basic_stringstream< char >::basic_stringstream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator>>() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.
For more information, see the iomanip header.

operator>>() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __istream_type &(* __pf )(__istream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::ws` and `std::dec` use these functions in constructs like `std::cin >> std::ws`.
For more information, see the iomanip header.

operator>>() [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            __streambuf_type * __sb)  [inherited]
```

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from this stream and inserted into the `__sb` streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.
References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            bool & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            float & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

operator>>() [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            int & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.

Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            short & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator>>() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned int & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned short & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            void *& __p)  [inline], [inherited]
```

Basic arithmetic extractors.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

peek()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
            void )  [inherited]
```

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, good() is false, returns traits::eof(). Otherwise reads but does not extract the next input character.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

putback()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
            char_type __c)  [inherited]
```

Unextracting a single character.

Parameters

_↩ _c	The character to push back into the input stream.

Returns

∗this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.
If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().
Referenced by std::operator<<().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type * std::basic_istringstream< _CharT, _Traits, _Alloc >::rdbuf () const  [inline],
[nodiscard]
```

Accessing the underlying buffer.

Returns

The current basic_stringbuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::p std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type > and std::basic_istream< _CharT, _Traits >::unget().

read()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction without delimiters.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into __*s* until one of the following happens:

- __*n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount.

readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction until the buffer is exhausted, but no more.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into __*s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::
and std::basic_ios< _CharT, _Traits >::setstate().

register_callback()

```
void std::ios_base::register_callback (
             event_callback __fn,
             int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
             off_type __off,
             ios_base::seekdir __dir)  [inherited]
```

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits and std::basic_ios< _CharT, _Traits >::setstate().

seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            pos_type __pos)  [inherited]
```

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),
std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(),
std::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::ope
std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >
std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_istringstream< _CharT, _Traits, _Alloc >::str () const &  [inline],
[nodiscard]
```

Copying out the string buffer.

Returns

`rdbuf()->str()`

str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_istringstream< _CharT, _Traits, _Alloc >::str (
            const __string_type & __s)  [inline]
```

Setting a new buffer.

Parameters

_↩ _s	The string to use as a new sequence.

Calls `rdbuf()->str(s)`.

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
            void )  [inherited]
```

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.
Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

sync_with_stdio()

```
bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/↩ libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellg()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
            void )  [inherited]
```

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unget()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
            void )  [inherited]
```

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.
If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sungetc().
Referenced by std::__detail::operator>>().

unsetf()

```
void std::ios_base::unsetf (
             fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
             char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_↩ _c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html
Referenced by std::endl(), std::basic_ios< char_type, traits_type >::fill(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::getline(), std::getline(), std::getline(), std::operator<<(), std::operator<<(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

xalloc()

```
int std::ios_base::xalloc () throw ( )   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

6.242.6 Member Data Documentation

_M_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount  [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see gcount().
Referenced by get(), get(), get(), get(), getline(), ignore(), ignore(), ignore(), putback(), read(), readsome(), and unget().

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```

Seek to end before each write.
Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

const openmode std::ios_base::ate [static], [inherited]
Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

const iostate std::ios_base::badbit [static], [inherited]
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_typ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

basefield

const fmtflags std::ios_base::basefield [static], [inherited]
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

const seekdir std::ios_base::beg [static], [inherited]
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

const openmode std::ios_base::binary [static], [inherited]
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩ filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static], [inherited]
Insert/extract `bool` in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

const seekdir std::ios_base::cur [static], [inherited]
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.
Referenced by std::dec(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::opera
and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>.

end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:
std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(),
std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, char_traits
std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type,
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(),
std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(),
std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_iostream< char_type, traits_type >::seekg(),
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```

Indicates all is well.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _T std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, char_traits< _CharT > >::tellg(), std::basic_istream< _Cha and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```

Open for input. Default for `ifstream` and fstream.

Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea std::basic_istringstream< char >::basic_istringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char > std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::p std::basic_streambuf< char_type, traits_type >::pubseekoff(), std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_streambuf< char_type, traits std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_streambuf< char_type, trait std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< char >::seekpos(), std::basic_filebuf< _CharT, _Traits > std::basic_stringbuf< char >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underfl std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::left(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x

oct

```
const fmtflags std::ios_base::oct  [static], [inherited]
```
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static], [inherited]
```
Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofs
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::o
std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::pubseekoff(),
std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_streambuf< char_type, traits_type >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_streambuf< char_type, traits_type >::seekpos(), std::basic_stringbuf< _Cha
std::basic_stringbuf< char >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsput

right

```
const fmtflags std::ios_base::right  [static], [inherited]
```
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).
Referenced by std::right().

scientific

```
const fmtflags std::ios_base::scientific  [static], [inherited]
```
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

```
const fmtflags std::ios_base::showbase  [static], [inherited]
```
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

```
const fmtflags std::ios_base::showpoint  [static], [inherited]
```
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

```
const fmtflags std::ios_base::showpos  [static], [inherited]
```
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static], [inherited]
```
Skips leading white space before certain input operations.

Referenced by std::noskipws(), std::__detail::operator>>(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x
and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static], [inherited]
```

Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

```
const fmtflags std::ios_base::uppercase  [static], [inherited]
```

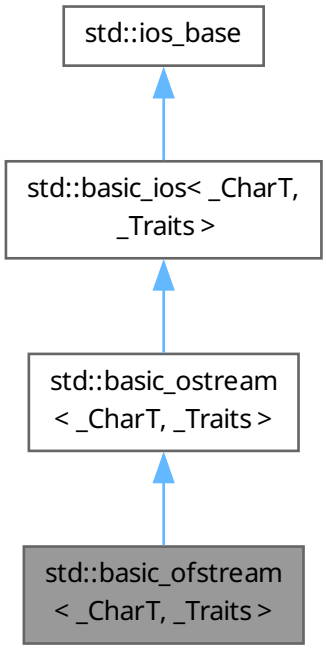
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following file:

- sstream

6.243 std::basic_ofstream< _CharT, _Traits > Class Template Reference

```
#include <fstream>
```

Inheritance diagram for std::basic_ofstream< _CharT, _Traits >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_filebuf< char_type, traits_type > **__filebuf_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< char_type, traits_type > **__ostream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef traits_type::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type

Public Member Functions

- basic_ofstream ()
- **basic_ofstream** (basic_ofstream &&__rhs)
- template<typename _Path, typename _Require = _If_fs_path<_Path>>
  basic_ofstream (const _Path &__s, ios_base::openmode __mode=ios_base::out)
- **basic_ofstream** (const basic_ofstream &)=delete
- basic_ofstream (const char *__s, ios_base::openmode __mode=ios_base::out)
- basic_ofstream (const std::string &__s, ios_base::openmode __mode=ios_base::out)
- ~basic_ofstream ()
- const locale & _M_getloc () const
- template<typename _ValueT>
  basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- void close ()
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- locale getloc () const
- bool good () const

- locale imbue (const locale &__loc)
- bool is_open ()
- bool **is_open** () const
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- template<typename _Path>
  _If_fs_path< _Path, void > open (const _Path &__s, ios_base::openmode __mode=ios_base::out)
- void open (const char *__s, ios_base::openmode __mode=ios_base::out)
- void open (const std::string &__s, ios_base::openmode __mode=ios_base::out)
- __ostream_type & operator<< (__streambuf_type *__sb)
- __ostream_type & operator<< (const void *__p)
- __ostream_type & **operator**<< (const volatile void *__p)
- __ostream_type & **operator**<< (nullptr_t)
- basic_ofstream & **operator=** (basic_ofstream &&__rhs)
- basic_ofstream & **operator=** (const basic_ofstream &)=delete
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- __filebuf_type * rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- __ostream_type & seekp (pos_type)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- void **swap** (basic_ofstream &__rhs)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)

- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- __ostream_type & write (const char_type *__s, streamsize __n)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal

- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT>
  __ostream_type & **_M_insert** (_ValueT __v)
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_ostream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

6.243.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_ofstream< _CharT, _Traits >

Controlling output for files.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This class supports reading from named files, using the inherited functions from std::basic_ostream. To control the associated sequence, an instance of std::basic_filebuf is used, which this page refers to as `sb`.

6.243.2 Member Typedef Documentation

__num_get_type

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_get_type  [inherited]
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

__e	One of the members of the event enum.
__b	Reference to the ios_base object.
__i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags  [inherited]
```

This is a bitmask type.
_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec

- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

iostate

`typedef _Ios_Iostate std::ios_base::iostate [inherited]`
This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

`typedef _Ios_Openmode std::ios_base::openmode [inherited]`
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir  [inherited]
```

This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

6.243.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

6.243.4 Constructor & Destructor Documentation

basic_ofstream() [1/4]

```
template<typename _CharT, typename _Traits>
std::basic_ofstream< _CharT, _Traits >::basic_ofstream ()  [inline]
```

Default constructor.
Initializes sb using its default constructor, and passes &sb to the base class initializer. Does not open any files (you haven't given it a filename to open).

basic_ofstream() [2/4]

```
template<typename _CharT, typename _Traits>
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
            const char * __s,
            ios_base::openmode __mode = ios_base::out)  [inline], [explicit]
```

Create an output file stream.

Parameters

__s	Null terminated string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

ios_base::out is automatically included in *__mode*.

basic_ofstream() [3/4]

```
template<typename _CharT, typename _Traits>
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
            const std::string & __s,
            ios_base::openmode __mode = ios_base::out)  [inline], [explicit]
```

Create an output file stream.

Parameters

__s	std::string specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::out` is automatically included in *__mode*.

basic_ofstream() [4/4]

```
template<typename _CharT, typename _Traits>
template<typename _Path, typename _Require = _If_fs_path<_Path>>
std::basic_ofstream< _CharT, _Traits >::basic_ofstream (
            const _Path & __s,
            ios_base::openmode __mode = ios_base::out)  [inline]
```

Create an output file stream.

Parameters

__s	filesystem::path specifying the filename.
__mode	Open file in specified mode (see std::ios_base).

`ios_base::out` is automatically included in *__mode*.

~basic_ofstream()

```
template<typename _CharT, typename _Traits>
std::basic_ofstream< _CharT, _Traits >::~basic_ofstream ()  [inline]
```

The destructor does nothing.
The file is closed by the filebuf object, not the formatting stream.

6.243.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
             iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::basic_fstream< char >::open(), std::basic_fstream< char >::
std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

close()

```
template<typename _CharT, typename _Traits>
void std::basic_ofstream< _CharT, _Traits >::close ()  [inline]
```

Close the file.
Calls `std::basic_filebuf::close()`. If that function fails, `failbit` is set in the stream's error state.

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
             const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::o

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<,
std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()  [inherited]
```

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.
References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits
Referenced by std::flush().

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(),
and std::__detail::operator>>().

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_ios< char_type, traits_type >::basic_ios(), std::basic_istream< char_type, traits_type >::basic_istream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ostream< char_type, traits_type >::basic_ostream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringstream< char >::basic_stringstream(),
std::basic_stringstream< char >::basic_stringstream(), and std::basic_stringstream< char >::basic_stringstream().

is_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_ofstream< _CharT, _Traits >::is_open ()  [inline], [nodiscard]
```

Wrapper to test for an open file.

Returns

`rdbuf()->is_open()`

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html

open() [1/3]

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, void > std::basic_ofstream< _CharT, _Traits >::open (
            const _Path & __s,
            ios_base::openmode __mode = ios_base::out)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file, as a filesystem::path.

__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode|out)`. If that function fails, `failbit` is set in the stream's error state.

open() [2/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ofstream< _CharT, _Traits >::open (
            const char * __s,
            ios_base::openmode __mode = ios_base::out)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(__s,__mode|out)`. If that function fails, `failbit` is set in the stream's error state.
Referenced by std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofstream(), and std::basic_ofstream< char >::open().

open() [3/3]

```
template<typename _CharT, typename _Traits>
void std::basic_ofstream< _CharT, _Traits >::open (
            const std::string & __s,
            ios_base::openmode __mode = ios_base::out)  [inline]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Calls `std::basic_filebuf::open(s,mode|out)`. If that function fails, `failbit` is set in the stream's error state.

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
```

bool std::basic_ios< _CharT, _Traits >::operator! () const [inline], [nodiscard], [inherited]
The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator<<() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ios_type &(* __pf )(__ios_type &)) [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ostream_type &(* __pf )(__ostream_type &)) [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            __streambuf_type * __sb) [inherited]
```

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

operator<<() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            bool __n) [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            const void * __p)  [inline], [inherited]
```

Pointer arithmetic inserters.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            float __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            int __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.

Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            short __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.
References basic_ostream(), std::ios_base::badbit, std::ostreambuf_iterator< _CharT, _Traits >::failed(), std::ios_base::goodbit, std::num_put< _CharT, _OutIter >::put(), std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator<<() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned int __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned short __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

put()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
            char_type __c)  [inherited]
```

Simple insertion.

Parameters

_↩ _c	The character to insert.

Returns

∗this

Tries to insert __c.

Note

This function is not overloaded on signed char and unsigned char.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits
Referenced by std::endl(), and std::ends().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void∗ associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_ofstream< _CharT, _Traits >::rdbuf () const  [inline], [nodiscard]
```

Accessing the underlying buffer.

Returns

The current basic_filebuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::p std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type > and std::basic_istream< _CharT, _Traits >::unget().

register_callback()

```
void std::ios_base::register_callback (
             event_callback __fn,
             int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.

__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

seekp() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            pos_type __pos)  [inherited]
```

Changing the current write position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),
std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(),
std::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::ope
std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >
std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

sync_with_stdio()

```
bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellp()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()  [inherited]
```

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.
References std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf()

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unsetf()

```
void std::ios_base::unsetf (
             fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
             char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Referenced by std::endl(), std::basic_ios< char_type, traits_type >::fill(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::getline(), std::getline(), std::getline(), std::operator<<(), std::operator<<(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

write()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
            const char_type * __s,
            streamsize __n)  [inherited]
```

Character string insertion.

Parameters

_↩ _s	The array to insert.
_↩ _n	Maximum number of characters to insert.

Returns

*this

Characters are copied from __s and inserted into the stream until one of the following happens:

- __n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

xalloc()

```
int std::ios_base::xalloc () throw ( )   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

6.243.6 Member Data Documentation

adjustfield

```
const fmtflags std::ios_base::adjustfield [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app [static], [inherited]
```

Seek to end before each write.
Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate [static], [inherited]
```

Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_typ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

basefield

```
const fmtflags std::ios_base::basefield [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

```
const seekdir std::ios_base::beg [static], [inherited]
```

Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

```
const openmode std::ios_base::binary [static], [inherited]
```

Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩ filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static], [inherited]
Insert/extract bool in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

const seekdir std::ios_base::cur [static], [inherited]
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

const fmtflags std::ios_base::dec [static], [inherited]
Converts integer input or generates integer output in decimal base.
Referenced by std::dec(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::opera and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>.

end

const seekdir std::ios_base::end [static], [inherited]
Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

const iostate std::ios_base::eofbit [static], [inherited]
Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >: std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Tra std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, char_traits std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

const iostate std::ios_base::failbit [static], [inherited]
Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char > std::basic_ofstream< char >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_iostream< char_type, traits_type >::seekg(),

std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```

Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```

Indicates all is well.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _T std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, char_traits< _CharT > >::tellg(), std::basic_istream< _Cha and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```

Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea std::basic_istringstream< char >::basic_istringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char > std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pb std::basic_streambuf< char_type, traits_type >::pubseekoff(), std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_streambuf< char_type, traits std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_streambuf< char_type, trait std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< char >::seekpos(), std::basic_filebuf< _CharT, _Traits > std::basic_stringbuf< char >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underfl std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

const fmtflags std::ios_base::internal [static], [inherited]
Adds fill characters at a designated internal point in certain generated output, or identical to right if no such point is designated.
Referenced by std::internal().

left

const fmtflags std::ios_base::left [static], [inherited]
Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::left(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x

oct

const fmtflags std::ios_base::oct [static], [inherited]
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

const openmode std::ios_base::out [static], [inherited]
Open for output. Default for ofstream and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofs
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::o
std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::pubseekoff(),
std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_streambuf< char_type, traits_type >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_streambuf< char_type, traits_type >::seekpos(), std::basic_stringbuf< _Cha
std::basic_stringbuf< char >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsput

right

const fmtflags std::ios_base::right [static], [inherited]
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).
Referenced by std::right().

scientific

const fmtflags std::ios_base::scientific [static], [inherited]
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

const fmtflags std::ios_base::showbase [static], [inherited]
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

```
const fmtflags std::ios_base::showpoint  [static], [inherited]
```

Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

```
const fmtflags std::ios_base::showpos  [static], [inherited]
```

Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static], [inherited]
```

Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x
and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static], [inherited]
```

Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

```
const fmtflags std::ios_base::uppercase  [static], [inherited]
```

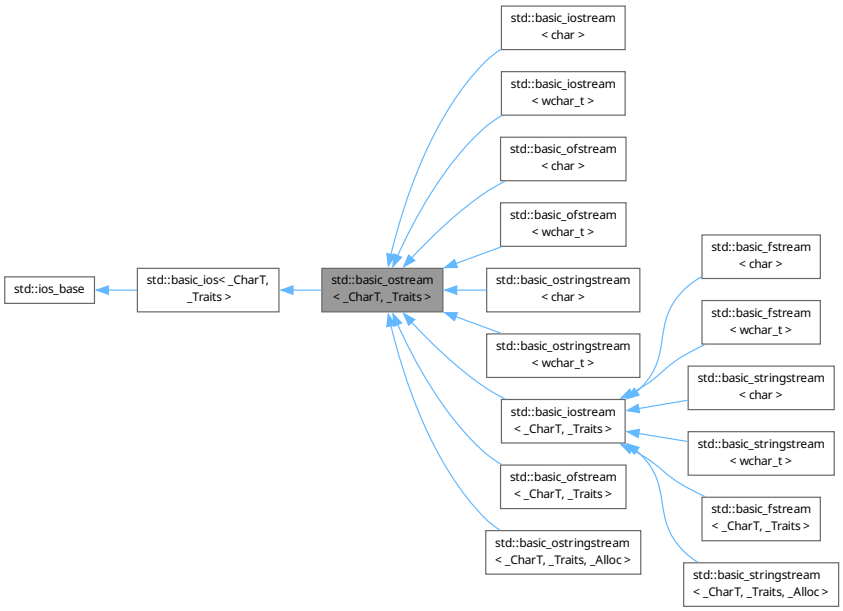
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following file:

- fstream

6.244 std::basic_ostream< _CharT, _Traits > Class Template Reference

```
#include <ostream.h>
```

Inheritance diagram for std::basic_ostream< _CharT, _Traits >:



Classes

- class sentry

Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< _CharT, _Traits > **__ostream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef _Traits::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef _Traits::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef _Traits::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type

Public Member Functions

- basic_ostream (__streambuf_type *__sb)
- virtual ~basic_ostream ()
- const locale & _M_getloc () const
- template<typename _ValueT>
  basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __ostream_type & operator<< (__streambuf_type *__sb)
- __ostream_type & operator<< (const void *__p)
- __ostream_type & **operator**<< (const volatile void *__p)
- __ostream_type & **operator**<< (nullptr_t)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf () const
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- __ostream_type & seekp (pos_type)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))

- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- __ostream_type & write (const char_type *__s, streamsize __n)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary

- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- **basic_ostream** (basic_iostream< _CharT, _Traits > &)
- **basic_ostream** (basic_ostream &&__rhs)
- **basic_ostream** (const basic_ostream &)=delete
- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT>
  __ostream_type & **_M_insert** (_ValueT __v)
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- basic_ostream & **operator=** (basic_ostream &&__rhs)
- basic_ostream & **operator=** (const basic_ostream &)=delete
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_ostream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

Friends

- class **sentry**

6.244.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_ostream< _CharT, _Traits >

Template class basic_ostream.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This is the base class for all output streams. It provides text formatting of all builtin types, and communicates with any class derived from basic_streambuf to do the actual output.

6.244.2 Member Typedef Documentation

__num_get_type

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_get_type  [inherited]
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```
The type of an event callback function.

Parameters

_↩ _e	One of the members of the event enum.
_↩ _b	Reference to the ios_base object.
_↩ _i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags  [inherited]
```
This is a bitmask type.
_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

iostate

`typedef _Ios_Iostate std::ios_base::iostate [inherited]`
This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

`typedef _Ios_Openmode std::ios_base::openmode [inherited]`
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

`typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`
This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

6.244.3 Member Enumeration Documentation

event

`enum std::ios_base::event [inherited]`
The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

6.244.4 Constructor & Destructor Documentation

basic_ostream()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::basic_ostream (
            __streambuf_type * __sb)  [inline], [explicit]
```

Base constructor.
This ctor is almost never called by the user directly, rather from derived classes' initialization lists, which pass a pointer to their own stream buffer.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and operator<<().

~basic_ostream()

```
template<typename _CharT, typename _Traits>
virtual std::basic_ostream< _CharT, _Traits >::~basic_ostream ()  [inline], [virtual]
```

Base destructor.
This does very little apart from providing a virtual base dtor.

6.244.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
            iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::basic_fstream< char >::open(), std::basic_fstream< char >::o
std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::o

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<,
std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()
```

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.
References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits
Referenced by std::flush().

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_ios< char_type, traits_type >::basic_ios(), std::basic_istream< char_type, traits_type >::basic_istream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ostream< char_type, traits_type >::basic_ostream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringstream< char >::basic_stringstream(),
std::basic_stringstream< char >::basic_stringstream(), and std::basic_stringstream< char >::basic_stringstream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)`
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator<<() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ios_type &(* __pf )(__ios_type &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ostream_type &(* __pf )(__ostream_type &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            __streambuf_type * __sb)
```

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If __sb is NULL, the stream will set failbit in its error state.
Characters are extracted from __sb and inserted into *this until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from __sb, which sets failbit in the error state

If the function inserts no characters, failbit is set.

operator<<() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            bool __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            const void * __p)  [inline]
```

Pointer arithmetic inserters.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            double __f)  [inline]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            float __f)  [inline]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            int __n)
```

Integer arithmetic inserters.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            ios_base &(* __pf )(ios_base &))  [inline]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _*n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long double __f)  [inline]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long long __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            short __n)
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.
References basic_ostream(), std::ios_base::badbit, std::ostreambuf_iterator< _CharT, _Traits >::failed(), std::ios_base::goodbit, std::num_put< _CharT, _OutIter >::put(), std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator<<() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned int __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long long __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned short __n)  [inline]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

put()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
            char_type __c)
```

Simple insertion.

Parameters

_↩ _c	The character to insert.

Returns

*this

Tries to insert __c.

Note

This function is not overloaded on signed char and unsigned char.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits
Referenced by std::endl(), and std::ends().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _*ix*	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf () const  [inline],
[nodiscard], [inherited]
```

Accessing the underlying buffer.

Returns

The current stream buffer.

This does not change the state of the stream.
Referenced by clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), imbue(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits > std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_ostream< _CharT, _Traits >::tel std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::| std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type > and std::basic_istream< _CharT, _Traits >::unget().

register_callback()

```
void std::ios_base::register_callback (
             event_callback __fn,
             int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

seekp() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
             off_type __off,
             ios_base::seekdir __dir)
```

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            pos_type __pos)
```

Changing the current write position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
             iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),
std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(),
std::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::ope
std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >
std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

sync_with_stdio()

```
bool std::ios_base::sync_with_stdio (
             bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellp()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()
```

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type(-1)` to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.
References std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf()

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
             basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unsetf()

```
void std::ios_base::unsetf (
             fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_↩ _c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html
Referenced by std::endl(), std::basic_ios< char_type, traits_type >::fill(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::getline(), std::getline(), std::getline(), std::operator<<(), std::operator<<(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

write()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
            const char_type * __s,
            streamsize __n)
```

Character string insertion.

Parameters

_↩ _s	The array to insert.
_↩ _n	Maximum number of characters to insert.

Returns

*this

Characters are copied from __s and inserted into the stream until one of the following happens:

- __n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

xalloc()

```
int std::ios_base::xalloc () throw ( )   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

6.244.6 Member Data Documentation

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of setf.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```

Seek to end before each write.
Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

const openmode std::ios_base::ate [static], [inherited]
Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

const iostate std::ios_base::badbit [static], [inherited]
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_typ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

basefield

const fmtflags std::ios_base::basefield [static], [inherited]
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

const seekdir std::ios_base::beg [static], [inherited]
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

const openmode std::ios_base::binary [static], [inherited]
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩ filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static], [inherited]
Insert/extract `bool` in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

const seekdir std::ios_base::cur [static], [inherited]
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

```
const fmtflags std::ios_base::dec  [static], [inherited]
```
Converts integer input or generates integer output in decimal base.
Referenced by std::dec(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::opera
and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>.

end

```
const seekdir std::ios_base::end  [static], [inherited]
```
Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

```
const iostate std::ios_base::eofbit  [static], [inherited]
```
Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:
std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(),
std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(),
std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, char_traits
std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit  [static], [inherited]
```
Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type,
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(),
std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(),
std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_iostream< char_type, traits_type >::seekg(),
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```
Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```
A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```

Indicates all is well.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _T std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, char_traits< _CharT > >::tellg(), std::basic_istream< _Cha and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```

Open for input. Default for `ifstream` and fstream.

Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea std::basic_istringstream< char >::basic_istringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char > std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::p std::basic_streambuf< char_type, traits_type >::pubseekoff(), std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_streambuf< char_type, traits std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_streambuf< char_type, trait std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< char >::seekpos(), std::basic_filebuf< _CharT, _Traits std::basic_stringbuf< char >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underfl std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.

Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).

Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::left(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x

oct

```
const fmtflags std::ios_base::oct  [static], [inherited]
```
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static], [inherited]
```
Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofs
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::o
std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::pubseekoff(),
std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_streambuf< char_type, traits_type >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_streambuf< char_type, traits_type >::seekpos(), std::basic_stringbuf< _Cha
std::basic_stringbuf< char >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsput

right

```
const fmtflags std::ios_base::right  [static], [inherited]
```
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).
Referenced by std::right().

scientific

```
const fmtflags std::ios_base::scientific  [static], [inherited]
```
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

```
const fmtflags std::ios_base::showbase  [static], [inherited]
```
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

```
const fmtflags std::ios_base::showpoint  [static], [inherited]
```
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

```
const fmtflags std::ios_base::showpos  [static], [inherited]
```
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static], [inherited]
```
Skips leading white space before certain input operations.

Referenced by std::noskipws(), std::__detail::operator>>(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x
and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static], [inherited]
```

Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

```
const fmtflags std::ios_base::uppercase  [static], [inherited]
```

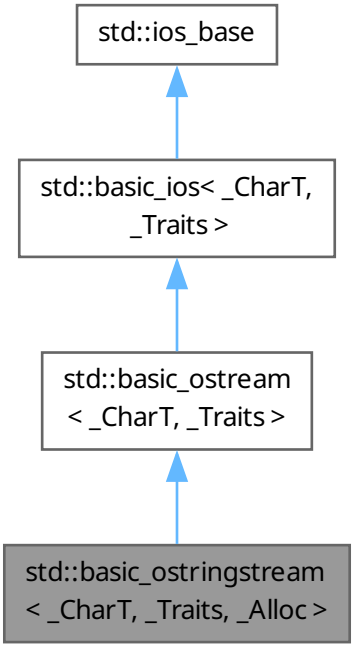
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following files:

- iosfwd.h
- ostream.h
- ostream.tcc

6.245 std::basic_ostringstream< _CharT, _Traits, _Alloc > Class Template Reference

```
#include <sstream>
```

Inheritance diagram for std::basic_ostringstream< _CharT, _Traits, _Alloc >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< char_type, traits_type > **__ostream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef basic_string< _CharT, _Traits, _Alloc > **__string_type**
- typedef basic_stringbuf< _CharT, _Traits, _Alloc > **__stringbuf_type**
- typedef _Alloc **allocator_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef traits_type::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > __num_get_type

Public Member Functions

- basic_ostringstream ()
- **basic_ostringstream** (__string_type &&__str, ios_base::openmode __mode=ios_base::out)
- **basic_ostringstream** (basic_ostringstream &&__rhs)
- basic_ostringstream (const __string_type &__str, ios_base::openmode __mode=ios_base::out)
- **basic_ostringstream** (const basic_ostringstream &)=delete
- template<typename _SAlloc>
  basic_ostringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, const allocator_type &__a)
- template<typename _SAlloc>
  basic_ostringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, ios_base::openmode __mode, const allocator_type &__a)
- template<typename _SAlloc>
  basic_ostringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, ios_base::openmode __↩ mode=ios_base::out)
- basic_ostringstream (ios_base::openmode __mode)
- **basic_ostringstream** (ios_base::openmode __mode, const allocator_type &__a)
- ~basic_ostringstream ()
- const locale & _M_getloc () const
- template<typename _ValueT>
  basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- locale getloc () const
- bool good () const
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __ostream_type & operator<< (__streambuf_type *__sb)
- __ostream_type & operator<< (const void *__p)
- __ostream_type & **operator**<< (const volatile void *__p)
- __ostream_type & **operator**<< (nullptr_t)
- basic_ostringstream & **operator=** (basic_ostringstream &&__rhs)
- basic_ostringstream & **operator=** (const basic_ostringstream &)=delete
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- __stringbuf_type * rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)

- __ostream_type & seekp (off_type, ios_base::seekdir)
- __ostream_type & seekp (pos_type)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- __string_type **str** () &&
- __string_type str () const &
- void **str** (__string_type &&__s)
- void str (const __string_type &__s)
- template<__allocator_like _SAlloc>
  basic_string< _CharT, _Traits, _SAlloc > **str** (const _SAlloc &__sa) const
- template<__allocator_like _SAlloc>
  void **str** (const basic_string< _CharT, _Traits, _SAlloc > &__s)
- void **swap** (basic_ostringstream &__rhs)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- basic_string_view< char_type, traits_type > **view** () const noexcept
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- __ostream_type & write (const char_type *__s, streamsize __n)

- operator bool () const
- bool operator! () const

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT>
  __ostream_type & **_M_insert** (_ValueT __v)
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept
- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_ostream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

6.245.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_ostringstream< _CharT, _Traits, _Alloc >

Controlling output for std::string.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

This class supports writing to objects of type std::basic_string, using the inherited functions from std::basic_ostream. To control the associated sequence, an instance of std::basic_stringbuf is used, which this page refers to as `sb`.

6.245.2 Member Typedef Documentation

__num_get_type

```
template<typename _CharT, typename _Traits>
typedef num_get<_CharT, istreambuf_iterator<_CharT, _Traits> > std::basic_ios< _CharT, _Traits
>::__num_get_type  [inherited]
```

These are non-standard types.

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_↩ _e	One of the members of the event enum.
_↩ _b	Reference to the ios_base object.
_↩ _i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags  [inherited]
```

This is a bitmask type.

_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal

- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield
- basefield
- floatfield

iostate

`typedef _Ios_Iostate std::ios_base::iostate [inherited]`
This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

`typedef _Ios_Openmode std::ios_base::openmode [inherited]`
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

```
typedef _Ios_Seekdir std::ios_base::seekdir  [inherited]
```

This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to SEEK_CUR in the C standard library.
- end, equivalent to SEEK_END in the C standard library.

6.245.3 Member Enumeration Documentation

event

```
enum std::ios_base::event  [inherited]
```

The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

6.245.4 Constructor & Destructor Documentation

basic_ostringstream() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream ()  [inline]
```

Default constructor starts with an empty string buffer.
Initializes sb using mode|out, and passes &sb to the base class initializer. Does not allocate any buffer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

basic_ostringstream() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream (
            ios_base::openmode __mode)  [inline], [explicit]
```

Starts with an empty string buffer.

Parameters

__mode	Whether the buffer can read, or write, or both.

ios_base::out is automatically included in *mode*.
Initializes sb using mode|out, and passes &sb to the base class initializer. Does not allocate any buffer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

basic_ostringstream() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::basic_ostringstream (
            const __string_type & __str,
            ios_base::openmode __mode = ios_base::out)  [inline], [explicit]
```

Starts with an existing string buffer.

Parameters

__str	A string to copy as a starting buffer.
__mode	Whether the buffer can read, or write, or both.

`ios_base::out` is automatically included in *mode*.
Initializes `sb` using *str* and `mode|out`, and passes `&sb` to the base class initializer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

~basic_ostringstream()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_ostringstream< _CharT, _Traits, _Alloc >::~basic_ostringstream ()  [inline]
```

The destructor does nothing.
The buffer is deallocated by the stringbuf object, not the formatting stream.

6.245.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
            iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::basic_fstream< char >::open(), std::basic_fstream< char >::
std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
            const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::o

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
            char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<,
std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()  [inherited]
```

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.
References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits
Referenced by std::flush().

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(),
and std::__detail::operator>>().

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_ios< char_type, traits_type >::basic_ios(), std::basic_istream< char_type, traits_type >::basic_istream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(),
std::basic_istringstream< char >::basic_istringstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ostream< char_type, traits_type >::basic_ostream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(),
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringstream< char >::basic_stringstream(),
std::basic_stringstream< char >::basic_stringstream(), and std::basic_stringstream< char >::basic_stringstream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of
`std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)`
Additional l10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator<<() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ostream_type &(* __pf )(__ostream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            __streambuf_type * __sb)  [inherited]
```

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.

If __sb is NULL, the stream will set failbit in its error state.
Characters are extracted from __sb and inserted into *this until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from __sb, which sets failbit in the error state

If the function inserts no characters, failbit is set.

operator<<() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            bool __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            const void * __p)  [inline], [inherited]
```

Pointer arithmetic inserters.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            float __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            int __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            short __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.
References basic_ostream(), std::ios_base::badbit, std::ostreambuf_iterator< _CharT, _Traits >::failed(), std::ios_base::goodbit, std::num_put< _CharT, _OutIter >::put(), std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator<<() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned int __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to perform numeric formatting.

operator<<() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned short __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

put()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
            char_type __c)  [inherited]
```

Simple insertion.

Parameters

_↩ _c	The character to insert.

Returns

*this

Tries to insert __c.

Note

This function is not overloaded on signed char and unsigned char.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits
Referenced by std::endl(), and std::ends().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type * std::basic_ostringstream< _CharT, _Traits, _Alloc >::rdbuf () const  [inline],
[nodiscard]
```

Accessing the underlying buffer.

Returns

The current basic_stringbuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::| std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type > and std::basic_istream< _CharT, _Traits >::unget().

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

seekp() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            pos_type __pos)  [inherited]
```

Changing the current write position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),
std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(),
std::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::ope
std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >
std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_ostringstream< _CharT, _Traits, _Alloc >::str () const &  [inline],
[nodiscard]
```

Copying out the string buffer.

Returns

```
rdbuf()->str()
```

str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_ostringstream< _CharT, _Traits, _Alloc >::str (
            const __string_type & __s)  [inline]
```

Setting a new buffer.

Parameters

_↩ _s	The string to use as a new sequence.

Calls `rdbuf()->str(s)`.

sync_with_stdio()

```
bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/↩ libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellp()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()  [inherited]
```

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.
References std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf()

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_↩ _c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html
Referenced by std::endl(), std::basic_ios< char_type, traits_type >::fill(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::getline(), std::getline(), std::getline(), std::operator<<(), std::operator<<(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

write()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
            const char_type * __s,
            streamsize __n)  [inherited]
```

Character string insertion.

Parameters

_↩ _s	The array to insert.
_↩ _n	Maximum number of characters to insert.

Returns

*this

Characters are copied from __s and inserted into the stream until one of the following happens:

- __n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

xalloc()

```
int std::ios_base::xalloc () throw ( )   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

6.245.6 Member Data Documentation

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of setf.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

const openmode std::ios_base::app [static], [inherited]
Seek to end before each write.
Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

const openmode std::ios_base::ate [static], [inherited]
Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

const iostate std::ios_base::badbit [static], [inherited]
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_typ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

basefield

const fmtflags std::ios_base::basefield [static], [inherited]
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

const seekdir std::ios_base::beg [static], [inherited]
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

const openmode std::ios_base::binary [static], [inherited]
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.↩filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static], [inherited]
Insert/extract `bool` in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

```
const seekdir std::ios_base::cur [static], [inherited]
```

Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

```
const fmtflags std::ios_base::dec [static], [inherited]
```

Converts integer input or generates integer output in decimal base.
Referenced by std::dec(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::opera and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>.

end

```
const seekdir std::ios_base::end [static], [inherited]
```

Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

```
const iostate std::ios_base::eofbit [static], [inherited]
```

Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:: std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Tra std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, char_traits std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit [static], [inherited]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char > std::basic_ofstream< char >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed [static], [inherited]
```

Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```
A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```
Indicates all is well.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _T std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, char_traits< _CharT > >::tellg(), std::basic_istream< _Cha and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```
Converts integer input or generates integer output in hexadecimal base.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```
Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea std::basic_istringstream< char >::basic_istringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char > std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::p std::basic_streambuf< char_type, traits_type >::pubseekoff(), std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_streambuf< char_type, traits std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_streambuf< char_type, trait std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< char >::seekpos(), std::basic_filebuf< _CharT, _Traits > std::basic_stringbuf< char >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underfl std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```
Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```
Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::left(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x

oct

```
const fmtflags std::ios_base::oct  [static], [inherited]
```
Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static], [inherited]
```
Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofs
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::o
std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::pubseekoff(),
std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_streambuf< char_type, traits_type >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_streambuf< char_type, traits_type >::seekpos(), std::basic_stringbuf< _Cha
std::basic_stringbuf< char >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsput

right

```
const fmtflags std::ios_base::right  [static], [inherited]
```
Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).
Referenced by std::right().

scientific

```
const fmtflags std::ios_base::scientific  [static], [inherited]
```
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

```
const fmtflags std::ios_base::showbase  [static], [inherited]
```
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

```
const fmtflags std::ios_base::showpoint  [static], [inherited]
```
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

```
const fmtflags std::ios_base::showpos  [static], [inherited]
```
Generates a + sign in non-negative generated numeric output.

Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static], [inherited]
```

Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x
and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static], [inherited]
```

Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static], [inherited]
```

Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

```
const fmtflags std::ios_base::uppercase  [static], [inherited]
```

Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following file:

- sstream

6.246 std::basic_regex< _Ch_type, _Rx_traits > Class Template Reference

`#include <regex>`

Public Types

- typedef regex_constants::syntax_option_type **flag_type**
- typedef traits_type::locale_type **locale_type**
- typedef traits_type::string_type **string_type**
- typedef _Rx_traits **traits_type**
- typedef _Ch_type **value_type**

Public Member Functions

- basic_regex () noexcept
- template<typename _FwdIter>
  basic_regex (_FwdIter __first, _FwdIter __last, flag_type __f=ECMAScript)
- basic_regex (basic_regex &&__rhs) noexcept=default
- basic_regex (const _Ch_type *__p, flag_type __f=ECMAScript)
- basic_regex (const _Ch_type *__p, std::size_t __len, flag_type __f=ECMAScript)
- basic_regex (const basic_regex &__rhs)=default
- template<typename _Ch_traits, typename _Ch_alloc>
  basic_regex (const std::basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, flag_type __f=ECMAScript)
- basic_regex (initializer_list< _Ch_type > __l, flag_type __f=ECMAScript)
- ~basic_regex ()

- template<typename _InputIterator>
  basic_regex & assign (_InputIterator __first, _InputIterator __last, flag_type __flags=ECMAScript)
- basic_regex & assign (basic_regex &&__rhs) noexcept
- basic_regex & assign (const _Ch_type *__p, flag_type __flags=ECMAScript)
- basic_regex & assign (const _Ch_type *__p, size_t __len, flag_type __flags=ECMAScript)
- basic_regex & assign (const basic_regex &__rhs) noexcept
- template<typename _Ch_traits, typename _Alloc>
  basic_regex & assign (const basic_string< _Ch_type, _Ch_traits, _Alloc > &__s, flag_type __flags=ECMAScript)
- basic_regex & assign (initializer_list< _Ch_type > __l, flag_type __flags=ECMAScript)
- flag_type flags () const noexcept
- locale_type getloc () const noexcept
- locale_type imbue (locale_type __loc)
- unsigned int mark_count () const noexcept
- basic_regex & operator= (basic_regex &&)=default
- basic_regex & operator= (const _Ch_type *__p)
- basic_regex & operator= (const basic_regex &)=default
- template<typename _Ch_traits, typename _Alloc>
  basic_regex & operator= (const basic_string< _Ch_type, _Ch_traits, _Alloc > &__s)
- basic_regex & operator= (initializer_list< _Ch_type > __l)
- void swap (basic_regex &__rhs) noexcept

Static Public Attributes

Constants

std [28.8.1](1)

- static constexpr flag_type **icase**
- static constexpr flag_type **nosubs**
- static constexpr flag_type **optimize**
- static constexpr flag_type **collate**
- static constexpr flag_type **ECMAScript**
- static constexpr flag_type **basic**
- static constexpr flag_type **extended**
- static constexpr flag_type **awk**
- static constexpr flag_type **grep**
- static constexpr flag_type **egrep**
- static constexpr flag_type **multiline**

Friends

- template<typename _Bp, typename _Ap, typename _Cp, typename _Rp>
  bool **__detail::__regex_algo_impl** (_Bp, _Bp, match_results< _Bp, _Ap > &, const basic_regex< _Cp, _Rp > &, regex_constants::match_flag_type, __detail::_RegexExecutorPolicy, bool)
- template<typename, typename, typename>
  class **__detail::_Executor**

Related Symbols

(Note that these are not member symbols.)

- template<typename _Ch_type, typename _Rx_traits>
  void swap (basic_regex< _Ch_type, _Rx_traits > &__lhs, basic_regex< _Ch_type, _Rx_traits > &__rhs) noexcept

6.246.1 Detailed Description

template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::basic_regex< _Ch_type, _Rx_traits >

A regular expression.
Specializations of this class template represent regular expressions constructed from sequences of character type _↩ Ch_type. Use the std::regex typedef for std::basic_regex<char>.
A character sequence passed to the constructor will be parsed according to the chosen grammar, and used to create a state machine representing the regular expression. The regex object can then be passed to algorithms such as std::regex_match to match sequences of characters.
The syntax_option_type flag passed to the constructor selects from one of the supported regular expression grammars. The default is ECMAScript and the others are basic, extended, awk, grep, and egrep, which are variations on POSIX regular expressions.

Since

C++11

6.246.2 Constructor & Destructor Documentation

basic_regex() [1/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex ()  [inline], [noexcept]
```

Constructs a basic regular expression that does not match any character sequence.

basic_regex() [2/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
            const _Ch_type * __p,
            flag_type __f = ECMAScript)  [inline], [explicit]
```

Constructs a basic regular expression from the sequence [__p, __p + char_traits<_Ch_type>::length(__p)) interpreted according to the flags in __f.

Parameters

_↩ _p	A pointer to the start of a C-style null-terminated string containing a regular expression.
_↩ _f	Flags indicating the syntax rules and options.

Exceptions

regex_error	if __p is not a valid regular expression.

basic_regex() [3/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
```

```
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
            const _Ch_type * __p,
            std::size_t __len,
            flag_type __f = ECMAScript)  [inline]
```

Constructs a basic regular expression from the sequence [p, p + len) interpreted according to the flags in `f`.

Parameters

__p	A pointer to the start of a string containing a regular expression.
__len	The length of the string containing the regular expression.
__f	Flags indicating the syntax rules and options.

Exceptions

regex_error	if `__p` is not a valid regular expression.

basic_regex() [4/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
            const basic_regex< _Ch_type, _Rx_traits > & __rhs)  [default]
```

Copy-constructs a basic regular expression.

Parameters

__rhs	A `regex` object.

basic_regex() [5/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
            basic_regex< _Ch_type, _Rx_traits > && __rhs)  [default], [noexcept]
```

Move-constructs a basic regular expression.

Parameters

__rhs	A `regex` object.

basic_regex() [6/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Ch_alloc>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
```

```
            const std::basic_string< _Ch_type, _Ch_traits, _Ch_alloc > & __s,
            flag_type __f = ECMAScript)  [inline], [explicit]
```

Constructs a basic regular expression from the string `s` interpreted according to the flags in `f`.

Parameters

_↩ _s	A string containing a regular expression.
_↩ _f	Flags indicating the syntax rules and options.

Exceptions

regex_error	if __s is not a valid regular expression.

basic_regex() [7/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _FwdIter>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
            _FwdIter __first,
            _FwdIter __last,
            flag_type __f = ECMAScript)  [inline]
```

Constructs a basic regular expression from the range [first, last) interpreted according to the flags in `f`.

Parameters

__first	The start of a range containing a valid regular expression.
__last	The end of a range containing a valid regular expression.
__f	The format flags of the regular expression.

Exceptions

regex_error	if [__first, __last) is not a valid regular expression.

basic_regex() [8/8]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::basic_regex (
            initializer_list< _Ch_type > __l,
```

```
            flag_type __f = ECMAScript)  [inline]
```

Constructs a basic regular expression from an initializer list.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer list.
↩ _↩ ↩ _↩ *f*	The format flags of the regular expression.

Exceptions

regex_error	if __l is not a valid regular expression.

~basic_regex()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
std::basic_regex< _Ch_type, _Rx_traits >::~basic_regex ()  [inline]
```

Destroys a basic regular expression.

6.246.3 Member Function Documentation

assign() [1/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _InputIterator>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
            _InputIterator __first,
            _InputIterator __last,
            flag_type __flags = ECMAScript)  [inline]
```

Assigns a new regular expression to a regex object.

Parameters

__first	The start of a range containing a valid regular expression.
__last	The end of a range containing a valid regular expression.
__flags	Syntax option flags.

Exceptions

regex_error	if p does not contain a valid regular expression pattern interpreted according to __flags. If regex↩ _error is thrown, the object remains unchanged.

assign() [2/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
            basic_regex< _Ch_type, _Rx_traits > && __rhs)  [inline], [noexcept]
```

Move-assigns one regular expression to another.

Parameters

__rhs	Another regular expression object.

assign() [3/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
            const _Ch_type * __p,
            flag_type __flags = ECMAScript)  [inline]
```

Assigns a new regular expression to a regex object from a C-style null-terminated string containing a regular expression pattern.

Parameters

__p	A pointer to a C-style null-terminated string containing a regular expression pattern.
__flags	Syntax option flags.

Exceptions

regex_error	if __p does not contain a valid regular expression pattern interpreted according to `__flags`. If regex_error is thrown, ∗this remains unchanged.

assign() [4/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
            const _Ch_type * __p,
            size_t __len,
            flag_type __flags = ECMAScript)  [inline]
```

Assigns a new regular expression to a regex object from a C-style string containing a regular expression pattern.

Parameters

__p	A pointer to a C-style string containing a regular expression pattern.
__len	The length of the regular expression pattern string.
__flags	Syntax option flags.

Exceptions

regex_error	if p does not contain a valid regular expression pattern interpreted according to __flags. If regex↩ _error is thrown, *this remains unchanged.

assign() [5/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
             const basic_regex< _Ch_type, _Rx_traits > & __rhs)  [inline], [noexcept]
```

Assigns one regular expression to another.

Parameters

__rhs	Another regular expression object.

Referenced by std::basic_regex< char >::basic_regex(), std::basic_regex< char >::assign(), std::basic_regex< char >::operator=(), std::basic_regex< char >::operator=(), and std::basic_regex< char >::operator=().

assign() [6/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Alloc>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
             const basic_string< _Ch_type, _Ch_traits, _Alloc > & __s,
             flag_type __flags = ECMAScript)  [inline]
```

Assigns a new regular expression to a regex object from a string containing a regular expression pattern.

Parameters

__s	A string containing a regular expression pattern.
__flags	Syntax option flags.

Exceptions

regex_error	if __s does not contain a valid regular expression pattern interpreted according to __flags. If regex_error is thrown, *this remains unchanged.

assign() [7/7]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::assign (
             initializer_list< _Ch_type > __l,
             flag_type __flags = ECMAScript)  [inline]
```

Assigns a new regular expression to a regex object.

Parameters

__l	An initializer list representing a regular expression.
__flags	Syntax option flags.

Exceptions

regex_error	if `__l` does not contain a valid regular expression pattern interpreted according to `__flags`. If regex_error is thrown, the object remains unchanged.

flags()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
flag_type std::basic_regex< _Ch_type, _Rx_traits >::flags () const  [inline], [noexcept]
```

Gets the flags used to construct the regular expression or in the last call to assign().

getloc()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
locale_type std::basic_regex< _Ch_type, _Rx_traits >::getloc () const  [inline], [noexcept]
```

Gets the locale currently imbued in the regular expression object.

imbue()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
locale_type std::basic_regex< _Ch_type, _Rx_traits >::imbue (
            locale_type __loc)  [inline]
```

Imbues the regular expression object with the given locale.

Parameters

__loc	A locale.

mark_count()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
unsigned int std::basic_regex< _Ch_type, _Rx_traits >::mark_count () const  [inline], [noexcept]
```

Gets the number of marked subexpressions within the regular expression.

operator=() [1/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::operator= (
            basic_regex< _Ch_type, _Rx_traits > && )  [default]
```

Move-assigns one regular expression to another.

operator=() [2/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::operator= (
            const _Ch_type * __p)  [inline]
```

Replaces a regular expression with a new one constructed from a C-style null-terminated string.

Parameters

_↩ _*p*	A pointer to the start of a null-terminated C-style string containing a regular expression.

operator=() [3/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::operator= (
            const basic_regex< _Ch_type, _Rx_traits > & )  [default]
```

Assigns one regular expression to another.

operator=() [4/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
template<typename _Ch_traits, typename _Alloc>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::operator= (
            const basic_string< _Ch_type, _Ch_traits, _Alloc > & __s)  [inline]
```

Replaces a regular expression with a new one constructed from a string.

Parameters

_↩ _*s*	A pointer to a string containing a regular expression.

operator=() [5/5]

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
basic_regex & std::basic_regex< _Ch_type, _Rx_traits >::operator= (
            initializer_list< _Ch_type > __l)  [inline]
```

Replaces a regular expression with a new one constructed from an initializer list.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer list.

Exceptions

regex_error	if __l is not a valid regular expression.

swap()

```
template<typename _Ch_type, typename _Rx_traits = regex_traits<_Ch_type>>
void std::basic_regex< _Ch_type, _Rx_traits >::swap (
            basic_regex< _Ch_type, _Rx_traits > & __rhs)  [inline], [noexcept]
```

Swaps the contents of two regular expression objects.

Parameters

__rhs	Another regular expression object.

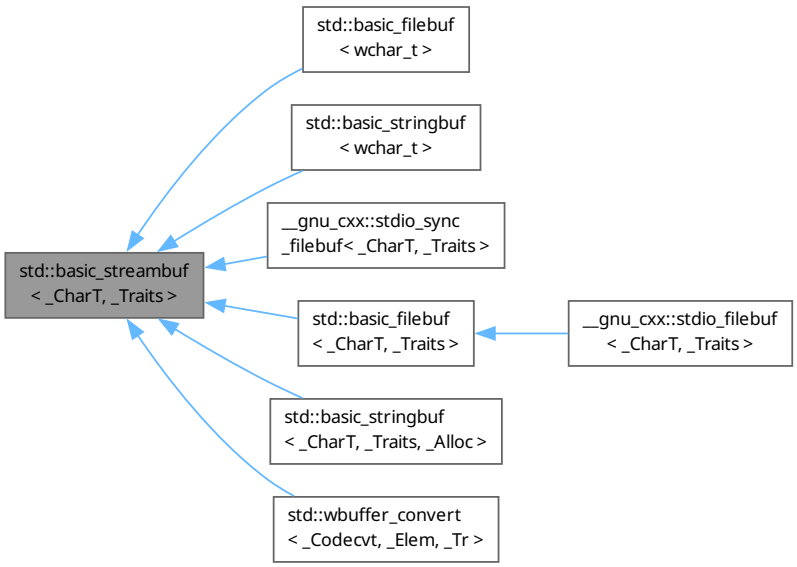
The documentation for this class was generated from the following file:

- regex.h

6.247 std::basic_streambuf< _CharT, _Traits > Class Template Reference

`#include <streambuf>`

Inheritance diagram for std::basic_streambuf< _CharT, _Traits >:



Public Types

- typedef _CharT char_type
- typedef _Traits traits_type
- typedef traits_type::int_type int_type
- typedef traits_type::pos_type pos_type
- typedef traits_type::off_type off_type

- typedef basic_streambuf< char_type, traits_type > __streambuf_type

Public Member Functions

- virtual ~basic_streambuf ()
- locale getloc () const
- streamsize in_avail ()
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)
- int_type sungetc ()

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- basic_streambuf ()
- **basic_streambuf** (const basic_streambuf &)
- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- basic_streambuf & **operator=** (const basic_streambuf &)
- virtual int_type overflow (int_type __c=traits_type::eof())
- virtual int_type pbackfail (int_type __c=traits_type::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type, ios_base::openmode=ios_base::in|ios_base::out)
- virtual basic_streambuf< char_type, _Traits > * setbuf (char_type *, streamsize)
- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)

- char_type * eback () const
- char_type * gptr () const
- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- locale _M_buf_locale
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end

Friends

- template<bool _IsMove, typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type **__copy_move_a2** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 *)
- streamsize **__copy_streambufs_eof** (basic_streambuf *__sbin, basic_streambuf *__sbout, bool &__ineof)
- void **__istream_extract** (istream &, char *, streamsize)
- template<typename _CharT2, typename _Distance>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, void >::__type **advance** (istreambuf_iterator< _CharT2 > &, _Distance)
- class **basic_ios< char_type, traits_type >**
- class **basic_istream< char_type, traits_type >**
- class **basic_ostream< char_type, traits_type >**
- template<typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 > >::__type **find** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)
- template<typename _CharT2, typename _Traits2, typename _Alloc>
  basic_istream< _CharT2, _Traits2 > & **getline** (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &, _CharT2)
- class **istreambuf_iterator< char_type, traits_type >**
- template<typename _CharT2, typename _Traits2, typename _Alloc>
  basic_istream< _CharT2, _Traits2 > & **operator>>** (basic_istream< _CharT2, _Traits2 > &, basic_string< _CharT2, _Traits2, _Alloc > &)
- class **ostreambuf_iterator< char_type, traits_type >**

6.247.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_streambuf< _CharT, _Traits >

The actual work of input and output (interface).

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.

This is a base class. Derived stream buffers each control a pair of character sequences: one for input, and one for output.
Section [27.5.1] of the standard describes the requirements and behavior of stream buffer classes. That section (three paragraphs) is reproduced here, for simplicity and accuracy.

1. Stream buffers can impose various constraints on the sequences they control. Some constraints are:

- The controlled input sequence can be not readable.
- The controlled output sequence can be not writable.
- The controlled sequences can be associated with the contents of other representations for character sequences, such as external files.
- The controlled sequences can support operations *directly* to or from associated sequences.
- The controlled sequences can impose limitations on how the program can read characters from a sequence, write characters to a sequence, put characters back into an input sequence, or alter the stream position.

2. Each sequence is characterized by three pointers which, if non-null, all point into the same `charT` array object. The array object represents, at any moment, a (sub)sequence of characters from the sequence. Operations performed on a sequence alter the values stored in these pointers, perform reads and writes directly to or from associated sequences, and alter *the stream position* and conversion state as needed to maintain this subsequence relationship. The three pointers are:
   - the *beginning pointer*, or lowest element address in the array (called *xbeg* here);
   - the *next pointer*, or next element address that is a current candidate for reading or writing (called *xnext* here);
   - the *end pointer*, or first element address beyond the end of the array (called *xend* here).
3. The following semantic constraints shall always apply for any set of three pointers for a sequence, using the pointer names given immediately above:
   - If *xnext* is not a null pointer, then *xbeg* and *xend* shall also be non-null pointers into the same `charT` array, as described above; otherwise, *xbeg* and *xend* shall also be null.
   - If *xnext* is not a null pointer and *xnext* $<$ *xend* for an output sequence, then a *write position* is available. In this case, $*$*xnext* shall be assignable as the next element to write (to put, or to store a character value, into the sequence).
   - If *xnext* is not a null pointer and *xbeg* $<$ *xnext* for an input sequence, then a *putback position* is available. In this case, *xnext*[-1] shall have a defined value and is the next (preceding) element to store a character that is put back into the input sequence.
   - If *xnext* is not a null pointer and *xnext*$<$ *xend* for an input sequence, then a *read position* is available. In this case, $*$*xnext* shall have a defined value and is the next element to read (to get, or to obtain a character value, from the sequence).

6.247.2 Member Typedef Documentation

__streambuf_type

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<char_type, traits_type> std::basic_streambuf< _CharT, _Traits >::__↩
streambuf_type
```

This is a non-standard type.

char_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_streambuf< _CharT, _Traits >::char_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

int_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::int_type std::basic_streambuf< _CharT, _Traits >::int_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

off_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::off_type std::basic_streambuf< _CharT, _Traits >::off_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

pos_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::pos_type std::basic_streambuf< _CharT, _Traits >::pos_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

traits_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_streambuf< _CharT, _Traits >::traits_type
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

6.247.3 Constructor & Destructor Documentation

~basic_streambuf()

```
template<typename _CharT, typename _Traits>
virtual std::basic_streambuf< _CharT, _Traits >::~basic_streambuf ()  [inline], [virtual]
```

Destructor deallocates no buffer space.

basic_streambuf()

```
template<typename _CharT, typename _Traits>
std::basic_streambuf< _CharT, _Traits >::basic_streambuf ()  [inline], [protected]
```

Base constructor.

Only called from derived constructors, and sets up all the buffer data to zero, including the pointers described in the basic_streambuf class description. Note that, as a result,

- the class starts with no read nor write positions available,
- this is not an error

6.247.4 Member Function Documentation

eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const  [inline], [protected]
```

Access to the get area.

These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _Cha std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const  [inline], [protected]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_streambuf< char_type, traits_type >::in_avail std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< char std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const  [inline], [protected]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sputc(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
            int __n)  [inline], [protected]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.

Referenced by std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_streambuf< char_type, traits_type >::uflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const  [inline]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.
Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const  [inline], [protected]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy
std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_stringbuf< _CharT,
std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< cha
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_streambuf< char_type, traits_type >::sgetc(),
std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< char >::showmanyc(), std::basic_streambuf< char_type, traits
std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_streambuf< char_type, traits_type >::uflow(),
std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(),
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and
xsgetn().

imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
            const locale & __loc)  [inline], [protected], [virtual]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, and std::basic_filebuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

in_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ()  [inline]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

overflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::overflow (
            int_type __c = traits_type::eof())  [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

_↩ _c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually __c, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if __c is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _T std::basic_stringbuf< char >, std::basic_stringbuf< wchar_t >, and std::wbuffer_convert< _Codecvt, _Elem, _Tr >.
Referenced by std::basic_streambuf< char_type, traits_type >::sputc(), and xsputn().

pbackfail()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::pbackfail (
            int_type __c = traits_type::eof())  [inline], [protected], [virtual]
```

Tries to back up the input sequence.

Parameters

_↩ _c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Tr std::basic_stringbuf< char >, and std::basic_stringbuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::sputbackc(), and std::basic_streambuf< char_type, traits_type >::sunget

pbase()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const  [inline], [protected]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, std::basic_stringbuf< char >::str(), std::basic_filebuf< _CharT, _Traits >::sync(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
            int __n)  [inline], [protected]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), and std::basic_streambuf< char_type, traits_type >::sputc().

pptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const  [inline], [protected]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_streambuf< char_t std::basic_filebuf< _CharT, _Traits >::sync(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
            const locale & __loc)  [inline]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline]
```

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
```

```
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
            char_type * __s,
            streamsize __n)  [inline]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ()  [inline]
```

Calls virtual sync function.
Referenced by std::basic_istream< _CharT, _Traits >::sync().

sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ()  [inline]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_streambuf< char_type, traits_type >::snextc().

seekoff()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
            off_type ,
            ios_base::seekdir ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [inline], [protected], [virtual]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Ti std::basic_stringbuf< char >, and std::basic_stringbuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubseekoff().

seekpos()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (
            pos_type ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [inline], [protected], [virtual]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Ti std::basic_stringbuf< char >, and std::basic_stringbuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubseekpos().

setbuf()

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf< char_type, _Traits > * std::basic_streambuf< _CharT, _Traits >::setbuf
(
            char_type * ,
            streamsize )  [inline], [protected], [virtual]
```

Manipulates the buffer.
Each derived class provides its own appropriate behavior. See the next-to-last paragraph of https://gcc.gnu.↩org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering for more on this function.

Note

Base class version does nothing, returns `this`.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, and std::basic_stringbuf< char >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubsetbuf().

setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
            char_type * __gbeg,
            char_type * __gnext,
            char_type * __gend)  [inline], [protected]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), std::basic_filebuf< char_type, traits_type >::_M_create
std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_filebuf< char_type, traits_type >::_M_set_buffer(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
            char_type * __pbeg,
            char_type * __pend)  [inline], [protected]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), and std::basic_filebuf< char_type, traits_type >::_M_s

sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ()  [inline]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::get(),
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ign
and std::basic_streambuf< char_type, traits_type >::snextc().

sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
```

```
        char_type * __s,
        streamsize __n)  [inline]
```

Entry point for xsgetn.

Parameters

_↩ _s	A buffer area.
_↩ _n	A count.

Returns xsgetn(__s,__n). The effect is to fill __s[0] through __s[__n-1] with characters from the input sequence, if possible.

showmanyc()

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf< _CharT, _Traits >::showmanyc ()  [inline], [protected],
[virtual]
```

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, and std::basic_stringbuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::in_avail().

snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ()  [inline]
```

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, char_traits< _CharT > >::get(), and std::basic_istream< _CharT, char_traits< _CharT > >::readsome().

sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
            char_type __c)  [inline]
```

Pushing characters back into the input stream.

Parameters

_↩ _c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but __c is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be __c.
Referenced by std::basic_istream< _CharT, _Traits >::putback().

sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
            char_type __c)  [inline]
```

Entry point for all single-character output functions.

Parameters

_↩ _c	A character to output.

Returns

__c, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.
Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow().

sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
            const char_type * __s,
            streamsize __n)  [inline]
```

Entry point for all single-character output functions.

Parameters

_↩ _s	A buffer read area.
_↩ _n	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __*s*[0] through __*s*[__n-1] to the output sequence, if possible.

sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ()  [inline]
```

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.
Referenced by std::basic_istream< _CharT, _Traits >::unget().

sync()

```
template<typename _CharT, typename _Traits>
virtual int std::basic_streambuf< _CharT, _Traits >::sync ()  [inline], [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, and std::wbuffer_convert< _Codecvt, _Elem, _Tr >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubsync().

uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ()  [inline], [protected], [virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.
Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.
Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc(), and xsgetn().

underflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::underflow ()  [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, std::basic_stringbuf< wcha and std::wbuffer_convert< _Codecvt, _Elem, _Tr >.
Referenced by std::basic_streambuf< char_type, traits_type >::sgetc(), and std::basic_streambuf< char_type, traits_type >::uflow().

xsgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsgetn (
             char_type * __s,
             streamsize __n)  [protected], [virtual]
```

Multiple character extraction.

Parameters

_↩ _*s*	A buffer area.
_↩ _*n*	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __*s*[0] through __*s*[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __*n* characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, and std::basic_filebuf< char_type, traits_type >.
References egptr(), gptr(), std::min(), and uflow().
Referenced by std::basic_streambuf< char_type, traits_type >::sgetn().

xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual]
```

Multiple character insertion.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __s[0] through __s[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, and std::basic_filebuf< char_type, traits_type >.
References epptr(), std::min(), overflow(), and pptr().
Referenced by std::basic_streambuf< char_type, traits_type >::sputn().

6.247.5 Member Data Documentation

_M_buf_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale  [protected]
```

Current locale setting.
Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

_M_in_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg  [protected]
```

Start of get area.

_M_in_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur  [protected]
```

Current read area.

_M_in_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end  [protected]
```

End of get area.

_M_out_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg  [protected]
```
Start of put area.

_M_out_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur  [protected]
```
Current put area.

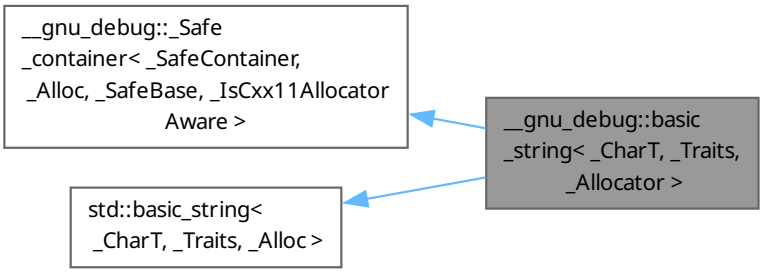
_M_out_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end  [protected]
```
End of put area.
The documentation for this class was generated from the following files:

- streambuf
- streambuf.tcc

6.248 __gnu_debug::basic_string< _CharT, _Traits, _Allocator > Class Template Reference

```
#include <string>
```
Inheritance diagram for __gnu_debug::basic_string< _CharT, _Traits, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< typename _Base::const_iterator, basic_string > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< typename _Base::iterator, basic_string > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**

- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- **basic_string** (_Base &&__base) noexcept
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  basic_string (_InputIterator __begin, _InputIterator __end, const _Allocator &__a=_Allocator())
- **basic_string** (basic_string &&)=default
- **basic_string** (basic_string &&__s, const _Allocator &__a) noexcept(std::is_nothrow_constructible< _Base, _↩ Base, const _Allocator & >::value)
- **basic_string** (const _Allocator &__a) noexcept
- **basic_string** (const _Base &__base)
- **basic_string** (const _CharT *__s, const _Allocator &__a=_Allocator())
- **basic_string** (const _CharT *__s, size_type __n, const _Allocator &__a=_Allocator())
- **basic_string** (const basic_string &)=default
- **basic_string** (const basic_string &__s, const _Allocator &__a)
- **basic_string** (const basic_string &__str, size_type __pos, size_type __n=_Base::npos, const _Allocator &__↩ a=_Allocator())
- **basic_string** (size_type __n, _CharT __c, const _Allocator &__a=_Allocator())
- **basic_string** (std::initializer_list< _CharT > __l, const _Allocator &__a=_Allocator())
- template<typename _Operation>
  constexpr void __resize_and_overwrite (size_type __n, _Operation __op)
- template<typename _Operation>
  void __resize_and_overwrite (size_type __n, _Operation __op)
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- template<typename _InputIterator>
  basic_string< _CharT, _Traits, _Alloc > & **_M_replace_dispatch** (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2, __false_type)
- template<typename _InIterator>
  _CharT * **_S_construct** (_InIterator __beg, _InIterator __end, const _Alloc &__a, forward_iterator_tag)
- template<typename _InputIterator>
  basic_string & **append** (_InputIterator __first, _InputIterator __last)
- basic_string & **append** (const _CharT *__s)
- basic_string & **append** (const _CharT *__s, size_type __n)
- basic_string & **append** (const basic_string &__str)
- basic_string & **append** (const basic_string &__str, size_type __pos, size_type __n)
- basic_string & **append** (size_type __n, _CharT __c)
- constexpr basic_string & append (const _CharT *__s, size_type __n)
- constexpr basic_string & append (const basic_string &__str)
- constexpr basic_string & append (const basic_string &__str, size_type __pos, size_type __n=npos)
- constexpr basic_string & append (initializer_list< _CharT > __l)
- constexpr basic_string & append (size_type __n, _CharT __c)
- template<typename _InputIterator>
  basic_string & **assign** (_InputIterator __first, _InputIterator __last)
- basic_string & **assign** (basic_string &&__x) noexcept(noexcept(std::declval< _Base & >().assign(std::move(_↩ _x))))
- basic_string & **assign** (const _CharT *__s)
- basic_string & **assign** (const _CharT *__s, size_type __n)

- basic_string & **assign** (const basic_string &__str, size_type __pos, size_type __n)
- basic_string & **assign** (const basic_string &__x)
- basic_string & **assign** (size_type __n, _CharT __c)
- basic_string & **assign** (std::initializer_list< _CharT > __l)
- constexpr basic_string & assign (basic_string &&__str) noexcept(_Alloc_traits::_S_nothrow_move())
- constexpr basic_string & assign (const _CharT *__s, size_type __n)
- constexpr basic_string & assign (const basic_string &__str)
- constexpr basic_string & assign (const basic_string &__str, size_type __pos, size_type __n=npos)
- constexpr basic_string & assign (size_type __n, _CharT __c)
- constexpr reference at (size_type __n)
- reference at (size_type __n)
- constexpr const_reference at (size_type __n) const
- const_reference at (size_type __n) const
- constexpr reference at (size_type __n)
- constexpr const_reference at (size_type __n) const
- constexpr const_reference back () const noexcept
- constexpr reference back () noexcept
- iterator **begin** ()
- const_iterator **begin** () const noexcept
- const _CharT * **c_str** () const noexcept
- constexpr size_type capacity () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** ()
- constexpr int **compare** (const _CharT *__s) const noexcept
- constexpr int compare (const basic_string &__str) const
- constexpr int compare (size_type __pos, size_type __n, const basic_string &__str) const
- int compare (size_type __pos, size_type __n, const basic_string &__str) const
- constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s) const
- constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__s) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__s, size_type __n2) const
- constexpr int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos) const
- int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos) const
- constexpr int compare (size_type __pos, size_type __n, const basic_string &__str) const
- constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s) const
- constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- constexpr int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos) const
- constexpr bool **contains** (_CharT __x) const noexcept
- bool **contains** (_CharT __x) const noexcept
- constexpr bool **contains** (basic_string_view< _CharT, _Traits > __x) const noexcept
- bool **contains** (basic_string_view< _CharT, _Traits > __x) const noexcept
- constexpr bool **contains** (const _CharT *__x) const noexcept
- bool **contains** (const _CharT *__x) const noexcept
- size_type **copy** (_CharT *__s, size_type __n, size_type __pos=0) const
- constexpr size_type copy (_CharT *__s, size_type __n, size_type __pos=0) const
- const_reverse_iterator **crbegin** () const noexcept

- const_reverse_iterator **crend** () const noexcept
- const _CharT * **data** () const noexcept
- constexpr _CharT * data () noexcept
- constexpr bool empty () const noexcept
- iterator **end** ()
- const_iterator **end** () const noexcept
- constexpr bool **ends_with** (_CharT __x) const noexcept
- bool **ends_with** (_CharT __x) const noexcept
- constexpr bool **ends_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- bool **ends_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- constexpr bool **ends_with** (const _CharT *__x) const noexcept
- bool **ends_with** (const _CharT *__x) const noexcept
- iterator **erase** (__const_iterator __first, __const_iterator __last)
- iterator **erase** (__const_iterator __position)
- basic_string & **erase** (size_type __pos=0, size_type __n=_Base::npos)
- constexpr iterator erase (__const_iterator __first, __const_iterator __last)
- constexpr iterator erase (__const_iterator __position)
- iterator erase (iterator __first, iterator __last)
- iterator erase (iterator __position)
- constexpr basic_string & erase (size_type __pos=0, size_type __n=npos)
- constexpr size_type find (_CharT __c, size_type __pos=0) const noexcept
- size_type find (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find** (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find** (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept

- constexpr size_type **find_first_of** (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find_first_of (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_first_of (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept
- size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__s, size_type __pos=_Base::npos) const noexcept
- constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos=npos) const noexcept
- constexpr size_type find_last_not_of (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_not_of (const basic_string &__str, size_type __pos=npos) const noexcept
- constexpr size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos=npos) const noexcept
- constexpr size_type find_last_not_of (const basic_string &__str, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__s, size_type __pos=_Base::npos) const noexcept
- constexpr size_type find_last_of (const _CharT *__s, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_of (const basic_string &__str, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_last_of (const _CharT *__s, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (const basic_string &__str, size_type __pos=npos) const noexcept
- constexpr const_reference front () const noexcept
- constexpr reference front () noexcept
- constexpr allocator_type get_allocator () const noexcept
- iterator **insert** (__const_iterator __p, _CharT __c)
- template<typename _InputIterator>
  iterator **insert** (__const_iterator __p, _InputIterator __first, _InputIterator __last)
- iterator **insert** (const_iterator __p, size_type __n, _CharT __c)
- iterator **insert** (const_iterator __p, std::initializer_list< _CharT > __l)
- basic_string & **insert** (size_type __pos, const _CharT *__s)
- basic_string & **insert** (size_type __pos, const _CharT *__s, size_type __n)
- basic_string & **insert** (size_type __pos, size_type __n, _CharT __c)
- basic_string & **insert** (size_type __pos1, const basic_string &__str)
- basic_string & **insert** (size_type __pos1, const basic_string &__str, size_type __pos2, size_type __n)
- constexpr iterator insert (__const_iterator __p, _CharT __c)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr iterator insert (const_iterator __p, _InputIterator __beg, _InputIterator __end)
- constexpr iterator insert (const_iterator __p, initializer_list< _CharT > __l)

- constexpr iterator insert (const_iterator __p, size_type __n, _CharT __c)
- iterator insert (iterator __p, _CharT __c)
- template<class _InputIterator>
  void insert (iterator __p, _InputIterator __beg, _InputIterator __end)
- void insert (iterator __p, initializer_list< _CharT > __l)
- void insert (iterator __p, size_type __n, _CharT __c)
- constexpr basic_string & insert (size_type __pos, const _CharT *__s)
- constexpr basic_string & insert (size_type __pos, const _CharT *__s, size_type __n)
- constexpr basic_string & insert (size_type __pos, size_type __n, _CharT __c)
- constexpr basic_string & insert (size_type __pos1, const basic_string &__str)
- constexpr basic_string & insert (size_type __pos1, const basic_string &__str, size_type __pos2, size_type __n=npos)
- constexpr size_type length () const noexcept
- constexpr size_type max_size () const noexcept
- basic_string & **operator+=** (_CharT __c)
- basic_string & **operator+=** (const _CharT *__s)
- basic_string & **operator+=** (const basic_string &__str)
- basic_string & **operator+=** (std::initializer_list< _CharT > __l)
- constexpr basic_string & operator+= (const basic_string &__str)
- basic_string & **operator=** (_CharT __c)
- basic_string & **operator=** (basic_string &&)=default
- basic_string & **operator=** (const _CharT *__s)
- basic_string & **operator=** (const basic_string &)=default
- basic_string & **operator=** (std::initializer_list< _CharT > __l)
- reference **operator[ ]** (size_type __pos)
- const_reference **operator[ ]** (size_type __pos) const noexcept
- constexpr reference operator[ ] (size_type __pos)
- constexpr const_reference operator[ ] (size_type __pos) const noexcept
- void **pop_back** ()
- void **push_back** (_CharT __c)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const noexcept
- template<typename _InputIterator>
  basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, _InputIterator __j1, _InputIterator __j2)
- basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const _CharT *__s)
- basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const _CharT *__s, size_type __n)
- basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const basic_string &__str)
- basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, size_type __n, _CharT __c)
- basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, std::initializer_list< _CharT > __l)
- basic_string & **replace** (size_type __pos, size_type __n1, const _CharT *__s)
- basic_string & **replace** (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)
- basic_string & **replace** (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- basic_string & **replace** (size_type __pos1, size_type __n1, const basic_string &__str)
- basic_string & **replace** (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, _CharT *__k1, _CharT *__k2)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const _CharT *__k1, const _CharT *__k2)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const _CharT *__s)

- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const _CharT *__s, size_type __n)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const basic_string &__str)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const_iterator __k1, const_iterator __k2)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, iterator __k1, iterator __k2)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, size_type __n, _CharT __c)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr basic_string & replace (const_iterator __i1, const_iterator __i2, _InputIterator __k1, _InputIterator __k2)
- constexpr basic_string & replace (const_iterator __i1, const_iterator __i2, initializer_list< _CharT > __l)
- basic_string & **replace** (iterator __i1, iterator __i2, _CharT *__k1, _CharT *__k2)
- template<class _InputIterator>
  basic_string & replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)
- basic_string & **replace** (iterator __i1, iterator __i2, const _CharT *__k1, const _CharT *__k2)
- basic_string & replace (iterator __i1, iterator __i2, const _CharT *__s)
- basic_string & replace (iterator __i1, iterator __i2, const _CharT *__s, size_type __n)
- basic_string & replace (iterator __i1, iterator __i2, const basic_string &__str)
- basic_string & **replace** (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)
- basic_string & replace (iterator __i1, iterator __i2, initializer_list< _CharT > __l)
- basic_string & **replace** (iterator __i1, iterator __i2, iterator __k1, iterator __k2)
- basic_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)
- constexpr basic_string & replace (size_type __pos, size_type __n, const basic_string &__str)
- constexpr basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s)
- constexpr basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)
- constexpr basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- constexpr basic_string & replace (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos)
- constexpr void reserve ()
- constexpr void reserve (size_type __res_arg)
- void reserve (size_type __res_arg)
- constexpr void reserve (size_type __res_arg)
- void **resize** (size_type __n)
- void **resize** (size_type __n, _CharT __c)
- constexpr void resize (size_type __n)
- constexpr void resize (size_type __n, _CharT __c)
- constexpr size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (const _CharT *__s, size_type __pos, size_type __n) const
- constexpr size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type **rfind** (const _CharT *__s, size_type __pos=_Base::npos) const
- constexpr size_type rfind (const _CharT *__s, size_type __pos=npos) const
- constexpr size_type rfind (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type rfind (const basic_string &__str, size_type __pos=npos) const noexcept
- constexpr size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type rfind (const _CharT *__s, size_type __pos=npos) const
- constexpr size_type rfind (const basic_string &__str, size_type __pos=npos) const noexcept
- void **shrink_to_fit** () noexcept
- constexpr size_type size () const noexcept
- constexpr bool **starts_with** (_CharT __x) const noexcept
- bool **starts_with** (_CharT __x) const noexcept

- constexpr bool **starts_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- bool **starts_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- constexpr bool **starts_with** (const _CharT *__x) const noexcept
- bool **starts_with** (const _CharT *__x) const noexcept
- basic_string **substr** (size_type __pos=0, size_type __n=_Base::npos) const
- constexpr basic_string **substr** (size_type __pos, size_type __n) &&
- constexpr basic_string **substr** (size_type __pos=0) &&
- constexpr basic_string substr (size_type __pos=0, size_type __n=npos) const
- constexpr basic_string_view< _CharT, _Traits > subview (size_type __pos=0, size_type __n=npos) const
- constexpr basic_string_view< _CharT, _Traits > subview (size_type __pos=0, size_type __n=npos) const
- void **swap** (basic_string &__x) noexcept(/*conditional */)
- constexpr void swap (basic_string &__s) noexcept

Static Public Attributes

- static const size_type npos

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

6.248.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
class __gnu_debug::basic_string< _CharT, _Traits, _Allocator >

Class std::basic_string with safety/checking/debug instrumentation.

6.248.2 Member Function Documentation

__resize_and_overwrite() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string< _CharT, _Traits, _Alloc >::__resize_and_overwrite (
            size_type __n,
            _Operation __op)  [constexpr], [inherited]
```

Non-standard version of resize_and_overwrite for C++11 and above.

__resize_and_overwrite() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string< _CharT, _Traits, _Alloc >::__resize_and_overwrite (
            size_type __n,
            _Operation __op)  [inherited]
```

Non-standard version of resize_and_overwrite for C++11 and above.

append() [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const _CharT * __s,
            size_type __n)  [inline], [constexpr], [inherited]
```

Append a C substring.

Parameters

_↩ _s	The C string to append.
_↩ _n	The number of characters to append.

Returns

Reference to this string.

append() [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr], [inherited]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

Referenced by std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::__cxx11::collate< _CharT >::do_transform(), std::basic_string< _CharT >::operator+=(), std::basic_string< _CharT >::operator+=(), std::basic_string< _CharT >::operator+=(), std::operator>>(), std::basic_string< _CharT, st and resize().

append() [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n = npos)  [inline], [constexpr], [inherited]
```

Append a substring.

Parameters

__str	The string to append.

__pos	Index of the first character of str to append.
__n	The number of characters to append.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *__pos* is not a valid index.

This function appends *__n* characters from *__str* starting at *__pos* to this string. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is appended.

append() [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            initializer_list< _CharT > __l)  [inline], [constexpr], [inherited]
```

Append an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to append.

Returns

Reference to this string.

append() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            size_type __n,
            _CharT __c)  [inline], [constexpr], [inherited]
```

Append multiple characters.

Parameters

_↩ *_n*	The number of characters to append.
_↩ *_c*	The character to use.

Returns

Reference to this string.

Appends __n copies of __c to this string.

assign() [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [constexpr], [noexcept],
[inherited]
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of *__str*. *__str* is a valid, but unspecified string.

assign() [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const _CharT * __s,
            size_type __n)  [inline], [constexpr], [inherited]
```

Set value to a C substring.

Parameters

_↩ _s	The C string to use.
_↩ _n	Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first *__n* characters of *__s*. If *__n* is is larger than the number of available characters in *__s*, the remainder of *__s* is used.

assign() [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr], [inherited]
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

Referenced by std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::operator=(), std::basic_string< _CharT >::operator=(), std::basic_string< _CharT >::operator=(), std::basic_string< _CharT >::operator=(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< char >::str() and std::basic_stringbuf< char >::str().

assign() [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n = npos)  [inline], [constexpr], [inherited]
```

Set value to a substring of a string.

Parameters

__str	The string to use.
__pos	Index of the first character of str.
__n	Number of characters to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *pos* is not a valid index.

This function sets this string to the substring of *__str* consisting of *__n* characters at *__pos*. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is used.

assign() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            size_type __n,
            _CharT __c)  [inline], [constexpr], [inherited]
```

Set value to multiple characters.

Parameters

_↩ _n	Length of the resulting string.
_↩ _c	The character to use.

Returns

Reference to this string.

This function sets the value of this string to __n copies of character __c.

at() [1/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string< _CharT, _Traits, _Allocator >::at (
            size_type __n)  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

at() [2/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string< _CharT, _Traits, _Allocator >::at (
            size_type __n)  [inline]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails. Success results in unsharing the string.

at() [3/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string< _CharT, _Traits, _Allocator >::at (
            size_type __n) const  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

at() [4/6]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string< _CharT, _Traits, _Allocator >::at (
            size_type __n) const  [inline]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

at() [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::at (
            size_type __n)  [inline], [nodiscard], [constexpr], [inherited]
```

Provides access to the data contained in the string.

Parameters

_↩ _*n*	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

at() [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::at (
            size_type __n) const  [inline], [nodiscard], [constexpr], [inherited]
```

Provides access to the data contained in the string.

Parameters

_↩ _*n*	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

back() [1/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string< _CharT, _Traits, _Allocator >::back () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

back() [2/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
reference std::basic_string< _CharT, _Traits, _Allocator >::back ()  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write reference to the data at the last element of the string.

capacity()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::capacity () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

compare() [1/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            const basic_string< _CharT, _Traits, _Allocator > & __str) const  [inline], [nodiscard],
[constexpr]
```

Compare to a string.

Parameters

__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *__str*, 0 if their values are equivalent, or > 0 if this string is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of size() and str.size(). The function then compares the two strings by calling traits::compare(data(), str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [2/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Allocator > & __str) const  [inline], [nodiscard],
[constexpr]
```

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [3/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Allocator > & __str) const  [inline]
```

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [4/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s) const  [inline], [nodiscard], [constexpr]
```

Compare substring to a C string.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *__s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *__s*. The function then compares the two string by calling traits::compare(substring.data(),__s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [5/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2) const  [inline], [nodiscard], [constexpr]
```

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Form a string from the first *__n2* characters of *__s*. Returns an integer < 0 if this substring is ordered before the string from *__s*, 0 if their values are equivalent, or

$>$ 0 if this substring is ordered after the string from __s. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and __n2. The function then compares the two strings by calling traits::↩ compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
NB: s must have at least n2 characters, '\0' has no special meaning.

compare() [6/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2) const  [inline]
```

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer $<$ 0, 0, or $>$ 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Form a string from the first *__n2* characters of *__s*. Returns an integer $<$ 0 if this substring is ordered before the string from *__s*, 0 if their values are equivalent, or $>$ 0 if this substring is ordered after the string from *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__n2*. The function then compares the two strings by calling traits::↩ compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
NB: s must have at least n2 characters, '\0' has no special meaning.

compare() [7/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos2,
            size_type __n2 = npos) const  [inline], [nodiscard], [constexpr]
```

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.

__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer < 0 if this substring is ordered before the substring of *__str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *__str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [8/12]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
int std::basic_string< _CharT, _Traits, _Allocator >::compare (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos2,
            size_type __n2 = npos) const  [inline]
```

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer < 0 if this substring is ordered before the substring of *__str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *__str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [9/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
```

```
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Alloc > & __str) const  [inline], [nodiscard],
[constexpr], [inherited]
```

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [10/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s) const  [inline], [nodiscard], [constexpr], [inherited]
```

Compare substring to a C string.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *__s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *__s*. The function then compares the two string by calling traits::compare(substring.data(),__s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [11/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
```

```
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2) const  [inline], [nodiscard], [constexpr], [inherited]
```

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Form a string from the first *__n2* characters of *__s*. Returns an integer < 0 if this substring is ordered before the string from *__s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__n2*. The function then compares the two strings by calling traits::↩ compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
NB: s must have at least n2 characters, '\0' has no special meaning.

compare() [12/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n2 = npos) const  [inline], [nodiscard], [constexpr], [inherited]
```

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer < 0 if this substring is ordered before the substring of *__str*,

0 if their values are equivalent, or > 0 if this substring is ordered after the substring of __*str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

copy()

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::copy (
            _CharT * __s,
            size_type __n,
            size_type __pos = 0) const  [constexpr], [inherited]
```

Copy substring into C string.

Parameters

__*s*	C string to copy value into.
__*n*	Number of characters to copy.
__*pos*	Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

std::out_of_range	If __pos > size().

Copies up to __*n* characters starting at __*pos* into the C string __*s*. If __*pos* is greater than size(), out_of_range is thrown.

data()

```
template<typename _CharT, typename _Traits, typename _Alloc>
_CharT * std::basic_string< _CharT, _Traits, _Alloc >::data ()  [inline], [nodiscard], [constexpr],
[noexcept], [inherited]
```

Return non-const pointer to contents.

This is a pointer to the character sequence held by the string. Modifying the characters in the sequence is allowed.

empty()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
bool std::basic_string< _CharT, _Traits, _Allocator >::empty () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

erase() [1/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
            __const_iterator __first,
            __const_iterator __last)  [inline], [constexpr], [inherited]
```

Remove a range of characters.

Parameters

__first	Iterator referencing the first character to remove.
__last	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

erase() [2/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
            __const_iterator __position)  [inline], [constexpr], [inherited]
```

Remove one character.

Parameters

__position	Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *__position* from this string. The value of the string doesn't change if an error is thrown.

erase() [3/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::iterator std::basic_string< _CharT, _Traits, _Alloc
>::erase (
            iterator __first,
            iterator __last)  [inherited]
```

Remove a range of characters.

Parameters

__first	Iterator referencing the first character to remove.
__last	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

erase() [4/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
            iterator __position)  [inline], [inherited]
```

Remove one character.

Parameters

__position	Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *__position* from this string. The value of the string doesn't change if an error is thrown.

erase() [5/5]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::erase (
            size_type __pos = 0,
            size_type __n = npos)  [inline], [constexpr], [inherited]
```

Remove characters.

Parameters

__pos	Index of first character to remove (default 0).
__n	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.

Removes *__n* characters from this string starting at *__pos*. The length of the string is reduced by *__n*. If there are < *__n* characters to remove, the remainder of the string is truncated. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::getline(), std::operator>>(), std::basic_string< _CharT >::pop_back(), and resize().

find() [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            _CharT __c,
            size_type __pos = 0) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find() [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            _CharT __c,
            size_type __pos = 0) const  [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find() [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
            _CharT __c,
            size_type __pos = 0) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.
References std::data(), npos, and std::size().
Referenced by std::basic_string< _CharT >::find(), std::basic_string< _CharT >::find(), std::basic_string< _CharT >::find(), and std::basic_string< _CharT >::find_first_of().

find() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
```

```
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

find_first_not_of() [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
            _CharT __c,
            size_type __pos = 0) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
            _CharT __c,
            size_type __pos = 0) const  [noexcept]
```

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from __s to consider.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from __s to consider.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
```

```
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_not_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
            _CharT __c,
            size_type __pos = 0) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from __*pos*, searches forward for a character other than __*c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_first_not_of() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character not in C substring.

Parameters

__*s*	C string containing characters to avoid.
__*pos*	Index of character to search from.
__*n*	Number of characters from __s to consider.

Returns

Index of first occurrence.

Starting from __*pos*, searches forward for a character not contained in the first __*n* characters of __*s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_first_not_of() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character not in C string.

Parameters

__*s*	C string containing characters to avoid.
__*pos*	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from __*pos*, searches forward for a character not contained in __*s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_first_not_of(), std::basic_string< _CharT >::find_first_not_of(), and std::basic_string< _CharT >::find_first_not_of().

find_first_of() [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            _CharT __c,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to find(__c, __pos).

find_first_of() [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            _CharT __c,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to find(__c, __pos).

find_first_of() [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C string.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_first_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            _CharT __c,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to find(__c, __pos).

find_first_of() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
```

```
>::find_first_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_first_of() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character of C string.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_first_of(), and std::basic_string< _CharT >::find_first_of().

find_last_not_of() [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            _CharT __c,
            size_type __pos = npos) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            _CharT __c,
            size_type __pos = npos) const  [noexcept]
```

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
```

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_not_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
            _CharT __c,
            size_type __pos = npos) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_last_not_of() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_last_not_of() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from __*pos*, searches backward for a character not contained in __*str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_last_not_of(), std::basic_string< _CharT >::find_last_not_of(), and std::basic_string< _CharT >::find_last_not_of().

find_last_of() [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            _CharT __c,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

Parameters

__*c*	Character to locate.
__*pos*	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from __*pos*, searches backward for __*c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to rfind(__c, __pos).

find_last_of() [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            _CharT __c,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character.

Parameters

__*c*	Character to locate.
__*pos*	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from __*pos*, searches backward for __*c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to rfind(__c, __pos).

find_last_of() [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C substring.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find last position of a character of C substring.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
```

```
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C string.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::find_last_of (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            _CharT __c,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to rfind(__c, __pos).

find_last_of() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character of C substring.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_last_of() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character of C string.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_last_of(), and std::basic_string< _CharT >::find_last_of().

front() [1/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const_reference std::basic_string< _CharT, _Traits, _Allocator >::front () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

front() [2/2]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
```

```
reference std::basic_string< _CharT, _Traits, _Allocator >::front ()  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read/write reference to the data at the first element of the string.

get_allocator()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
allocator_type std::basic_string< _CharT, _Traits, _Allocator >::get_allocator () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Return copy of allocator used to construct this string.

insert() [1/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            __const_iterator __p,
            _CharT __c)  [inline], [constexpr], [inherited]
```

Insert one character.

Parameters

_↩ _p	Iterator referencing position in string to insert at.
_↩ _c	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts character *__c* at position referenced by *__p*. If adding character causes the length to exceed max_size(), length_error is thrown. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [2/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            const_iterator __p,
            _InputIterator __beg,
            _InputIterator __end)  [inline], [constexpr], [inherited]
```

Insert a range of characters.

Parameters

__p	Const_iterator referencing location in string to insert at.
__beg	Start of range.
__end	End of range.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts characters in range [beg,end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [3/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            const_iterator __p,
            initializer_list< _CharT > __l)  [inline], [constexpr], [inherited]
```

Insert an initializer_list of characters.

Parameters

_↩ _p	Iterator referencing location in string to insert at.
_↩ _l	The initializer_list of characters to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

insert() [4/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            const_iterator __p,
            size_type __n,
            _CharT __c)  [inline], [constexpr], [inherited]
```

Insert multiple characters.

Parameters

_↩ _p	Const_iterator referencing location in string to insert at.
_↩ _n	Number of characters to insert
_↩ _c	The character to insert.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts __n copies of character __c starting at the position referenced by iterator __p. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), and std::basic_string< _CharT >::insert().

insert() [5/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            iterator __p,
            _CharT __c)  [inline], [inherited]
```

Insert one character.

Parameters

_↩ _p	Iterator referencing position in string to insert at.
_↩ _c	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts character __c at position referenced by __p. If adding character causes the length to exceed max_size(), length_error is thrown. If __p is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [6/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
             iterator __p,
             _InputIterator __beg,
             _InputIterator __end)  [inline], [inherited]
```

Insert a range of characters.

Parameters

__p	Iterator referencing location in string to insert at.
__beg	Start of range.
__end	End of range.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts characters in range [__beg,__end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [7/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
             iterator __p,
             initializer_list< _CharT > __l)  [inline], [inherited]
```

Insert an initializer_list of characters.

Parameters

_↩ _p	Iterator referencing location in string to insert at.
_↩ _l	The initializer_list of characters to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

insert() [8/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
```

```
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
            iterator __p,
            size_type __n,
            _CharT __c)  [inline], [inherited]
```

Insert multiple characters.

Parameters

_↩ _p	Iterator referencing location in string to insert at.
_↩ _n	Number of characters to insert
_↩ _c	The character to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts *__n* copies of character *__c* starting at the position referenced by iterator *__p*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [9/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            const _CharT * __s)  [inline], [constexpr], [inherited]
```

Insert a C string.

Parameters

__pos	Position in string to insert at.
__s	The C string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos* is beyond the end of this string.

Inserts the first *n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [10/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            const _CharT * __s,
            size_type __n)  [inline], [constexpr], [inherited]
```

Insert a C substring.

Parameters

__pos	Position in string to insert at.
__s	The C string to insert.
__n	The number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts the first *__n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [11/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            size_type __n,
            _CharT __c)  [inline], [constexpr], [inherited]
```

Insert multiple characters.

Parameters

__pos	Index in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts *__n* copies of character *__c* starting at index *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* > length(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [12/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos1,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr], [inherited]
```

Insert value of a string.

Parameters

__pos1	Position in string to insert at.
__str	The string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts value of *__str* starting at *__pos1*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [13/13]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n = npos)  [inline], [constexpr], [inherited]
```

Insert a substring.

Parameters

__pos1	Position in string to insert at.
__str	The string to insert.

__pos2	Start of characters in str to insert.
__n	Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos1* > size() or *__pos2* > *str.size()*.

Starting at *pos1*, insert *__n* character of *__str* beginning with *__pos2*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos1* is beyond the end of this string or *__pos2* is beyond the end of *__str*, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

length()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::length () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

max_size()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::max_size () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the size() of the largest possible string.

operator+=()

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr], [inherited]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

operator[]() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
            size_type __pos)  [inline], [nodiscard], [constexpr], [inherited]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

operator[]() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
            size_type __pos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)
Referenced by std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::front(), std::basic_string< _CharT >::front(), and std::basic_string< _CharT >::front().

replace() [1/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            const _CharT * __s)  [inline], [constexpr], [inherited]
```

Replace range of characters with C string.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.

_↩ _s	C string value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [2/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            const _CharT * __s,
            size_type __n)  [inline], [constexpr], [inherited]
```

Replace range of characters with C substring.

Parameters

_↩ _*i1*	Iterator referencing start of range to replace.
_↩ _*i2*	Iterator referencing end of range to replace.
_↩ _*s*	C string value to insert.
_↩ _*n*	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the first __*n* characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [3/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
```

```
__const_iterator __i1,
__const_iterator __i2,
const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr], [inherited]
```

Replace range of characters with string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__str	String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the value of *__str* is inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [4/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
             __const_iterator __i1,
             __const_iterator __i2,
             size_type __n,
             _CharT __c)  [inline], [constexpr], [inherited]
```

Replace range of characters with multiple characters.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _n	Number of characters to insert.
_↩ _c	Character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, __*n* copies of __*c* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [5/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
             const_iterator __i1,
             const_iterator __i2,
             _InputIterator __k1,
             _InputIterator __k2)  [inline], [constexpr], [inherited]
```

Replace range of characters with range.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__k1	Iterator referencing start of range to insert.
__k2	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [6/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
             const_iterator __i1,
             const_iterator __i2,
             initializer_list< _CharT > __l)  [inline], [constexpr], [inherited]
```

Replace range of characters with initializer_list.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _l	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [7/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
             iterator __i1,
             iterator __i2,
             _InputIterator __k1,
             _InputIterator __k2)  [inline], [inherited]
```

Replace range of characters with range.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__k1	Iterator referencing start of range to insert.
__k2	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [8/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            const _CharT * __s)  [inline], [inherited]
```

Replace range of characters with C string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__s	C string value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the characters of *__s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [9/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            const _CharT * __s,
            size_type __n)  [inline], [inherited]
```

Replace range of characters with C substring.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _s	C string value to insert.
_↩ _n	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the first *__n* characters of *__s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [10/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [inherited]
```

Replace range of characters with string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__str	String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the value of *__str* is inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [11/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            initializer_list< _CharT > __l)  [inline], [inherited]
```

Replace range of characters with initializer_list.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _l	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [12/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            size_type __n,
            _CharT __c)  [inline], [inherited]
```

Replace range of characters with multiple characters.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _n	Number of characters to insert.

_↩ _c	Character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, __*n* copies of __*c* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [13/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr], [inherited]
```

Replace characters with value from another string.

Parameters

__pos	Index of first character to replace.
__n	Number of characters to be replaced.
__str	String to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos+__n) from this string. In place, the value of __*str* is inserted. If __*pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::basic_string< _CharT >::append(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), and std::basic_string< _CharT >::replace().

replace() [14/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n1,
            const _CharT * __s)  [inline], [constexpr], [inherited]
```

Replace characters with value of a C string.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the characters of *__s* are inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [15/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::replace
(
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2)  [inline], [constexpr], [inherited]
```

Replace characters with value of a C substring.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.
__n2	Number of characters from *s* to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos1* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the first *__n2* characters of *__s* are inserted, or all of *__s* if *__n2* is too large. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [16/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n1,
            size_type __n2,
            _CharT __c)  [inline], [constexpr], [inherited]
```

Replace characters with multiple characters.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__n2	Number of characters to insert.
__c	Character to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, *__n2* copies of *__c* are inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [17/17]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos1,
            size_type __n1,
```

```
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n2 = npos)  [inline], [constexpr], [inherited]
```

Replace characters with value from another string.

Parameters

__pos1	Index of first character to replace.
__n1	Number of characters to be replaced.
__str	String to insert.
__pos2	Index of first character of str to use.
__n2	Number of characters from str to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size() or *__pos2* > __str.size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos1,__pos1 + n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

reserve() [1/4]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
void std::basic_string< _CharT, _Traits, _Allocator >::reserve ()  [constexpr]
```

Equivalent to shrink_to_fit().

reserve() [2/4]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
void std::basic_string< _CharT, _Traits, _Allocator >::reserve (
            size_type __res_arg)  [constexpr]
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

reserve() [3/4]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
void std::basic_string< _CharT, _Traits, _Allocator >::reserve (
            size_type __res_arg)
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

reserve() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve (
            size_type __res_arg)  [constexpr], [inherited]
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.
References capacity(), get_allocator(), and std::size().
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:

std::tr2::operator>>(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

resize() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
            size_type __n)  [inline], [constexpr], [inherited]
```

Resizes the string to the specified number of characters.

Parameters

_↩ _n	Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

resize() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
            size_type __n,
            _CharT __c)  [constexpr], [inherited]
```

Resizes the string to the specified number of characters.

Parameters

_↩ _n	Number of characters the string should contain.
_↩ _c	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to __c.
References append(), erase(), and size().
Referenced by std::basic_string< _CharT >::resize().

rfind() [1/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
            _CharT __c,
            size_type __pos = npos) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

rfind() [2/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
            _CharT __c,
            size_type __pos = npos) const  [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

rfind() [3/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [4/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find last position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [5/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr]
```

Find last position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [6/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
```

```
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [7/11]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::rfind (
            const basic_string< _CharT, _Traits, _Allocator > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [8/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
            _CharT __c,
            size_type __pos = npos) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from __pos, searches backward for __c within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

rfind() [9/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from __pos, searches backward for the first __n characters in __s within this string. If found, returns the index where it begins. If not found, returns npos.
References std::min(), npos, and std::size().

rfind() [10/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [inherited]
```

Find last position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from __pos, searches backward for the value of __s within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [11/11]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept], [inherited]
```

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_last_of(), std::basic_string< _CharT >::rfind(), std::basic_string< _CharT >::rfind(), std::basic_string< _CharT >::rfind(), and std::basic_string< _CharT >::rfind().

size()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
size_type std::basic_string< _CharT, _Traits, _Allocator >::size () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

substr()

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (
            size_type __pos = 0,
            size_type __n = npos) const  [inline], [nodiscard], [constexpr], [inherited]
```

Get a substring.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

std::out_of_range	If __pos > size().

Construct and return a new string using the __*n* characters starting at __*pos*. If the string is too short, use the remainder of the characters. If __*pos* is beyond the end of the string, out_of_range is thrown.

subview() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string_view< _CharT, _Traits > std::basic_string< _CharT, _Traits, _Alloc >::subview (
            size_type __pos = 0,
            size_type __n = npos) const  [inline], [nodiscard], [constexpr], [inherited]
```

Get a subview.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in subview (default remainder).

Returns

The subview.

Exceptions

std::out_of_range	If __pos > size().

Construct and return a subview using the __n characters starting at __pos. If the string is too short, use the remainder of the characters. If __pos is beyond the end of the string, out_of_range is thrown.

subview() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string_view< _CharT, _Traits > std::basic_string< _CharT, _Traits, _Alloc >::subview (
            size_type __pos = 0,
            size_type __n = npos) const  [inline], [nodiscard], [constexpr], [inherited]
```

Get a subview.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in subview (default remainder).

Returns

The subview.

Exceptions

std::out_of_range	If __pos > size().

Construct and return a subview using the __n characters starting at __pos. If the string is too short, use the remainder of the characters. If __pos is beyond the end of the string, out_of_range is thrown.

swap()

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::swap (
            basic_string< _CharT, _Traits, _Alloc > & __s)  [constexpr], [noexcept], [inherited]
```

Swap contents with another string.

Parameters

_↩ _s	String to swap with.

Exchanges the contents of this string with that of __*s* in constant time.
References basic_string(), and get_allocator().
Referenced by std::basic_string< _CharT >::assign(), and std::basic_string< _CharT >::operator=().

6.248.3 Member Data Documentation

npos

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator =
std::allocator<_CharT>>
const size_type std::basic_string< _CharT, _Traits, _Allocator >::npos  [static]
```

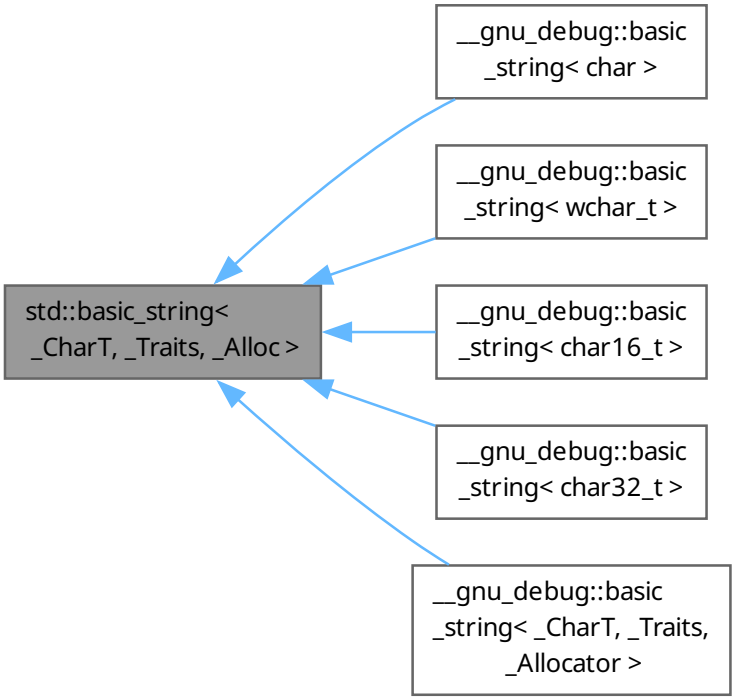
Value returned by various member functions when they fail.
The documentation for this class was generated from the following file:

- debug/string

6.249 std::basic_string< _CharT, _Traits, _Alloc > Class Template Reference

```
#include <string>
```

Inheritance diagram for std::basic_string< _CharT, _Traits, _Alloc >:



Public Types

- typedef _Char_alloc_type **allocator_type**
- typedef _Alloc **allocator_type**
- typedef __gnu_cxx::__normal_iterator< const_pointer, basic_string > **const_iterator**
- typedef __gnu_cxx::__normal_iterator< const_pointer, basic_string > **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _CharT_alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef const value_type & **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Alloc_traits::difference_type **difference_type**
- typedef _CharT_alloc_traits::difference_type **difference_type**
- typedef __gnu_cxx::__normal_iterator< pointer, basic_string > **iterator**
- typedef __gnu_cxx::__normal_iterator< pointer, basic_string > **iterator**
- typedef _Alloc_traits::pointer **pointer**
- typedef _CharT_alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef value_type & **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**

- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Alloc_traits::size_type **size_type**
- typedef _CharT_alloc_traits::size_type **size_type**
- typedef _Traits **traits_type**
- typedef _Traits **traits_type**
- typedef _Traits::char_type **value_type**
- typedef _Traits::char_type **value_type**

Public Member Functions

- basic_string () noexcept
- constexpr basic_string () noexcept(/*conditional */)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr basic_string (_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())
- template<class _InputIterator>
  basic_string (_InputIterator __beg, _InputIterator __end, const _Alloc &__a=_Alloc())
- constexpr basic_string (basic_string &&__str) noexcept
- basic_string (basic_string &&__str) noexcept
- **basic_string** (basic_string &&__str, const _Alloc &__a)
- constexpr **basic_string** (basic_string &&__str, const _Alloc &__a) noexcept(_Alloc_traits::_S_always_equal())
- constexpr **basic_string** (basic_string &&__str, size_type __pos, const _Alloc &__a=_Alloc())
- constexpr **basic_string** (basic_string &&__str, size_type __pos, size_type __n, const _Alloc &__a=_Alloc())
- basic_string (const _Alloc &__a)
- constexpr basic_string (const _Alloc &__a) noexcept
- template<typename = _RequireAllocator<_Alloc>>
  constexpr basic_string (const _CharT *__s, const _Alloc &__a=_Alloc())
- template<typename = _RequireAllocator<_Alloc>>
  basic_string (const _CharT *__s, const _Alloc &__a=_Alloc())
- constexpr basic_string (const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())
- basic_string (const _CharT *__s, size_type __n, const _Alloc &__a=_Alloc())
- constexpr basic_string (const basic_string &__str)
- basic_string (const basic_string &__str)
- constexpr **basic_string** (const basic_string &__str, const _Alloc &__a)
- **basic_string** (const basic_string &__str, const _Alloc &__a)
- constexpr basic_string (const basic_string &__str, size_type __pos, const _Alloc &__a=_Alloc())
- basic_string (const basic_string &__str, size_type __pos, const _Alloc &__a=_Alloc())
- constexpr basic_string (const basic_string &__str, size_type __pos, size_type __n)
- basic_string (const basic_string &__str, size_type __pos, size_type __n)
- constexpr basic_string (const basic_string &__str, size_type __pos, size_type __n, const _Alloc &__a)
- basic_string (const basic_string &__str, size_type __pos, size_type __n, const _Alloc &__a)
- constexpr basic_string (initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())
- basic_string (initializer_list< _CharT > __l, const _Alloc &__a=_Alloc())
- **basic_string** (nullptr_t)=delete
- **basic_string** (nullptr_t)=delete
- template<typename = _RequireAllocator<_Alloc>>
  constexpr basic_string (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())
- basic_string (size_type __n, _CharT __c, const _Alloc &__a=_Alloc())
- constexpr ~basic_string ()
- ~basic_string () noexcept
- template<typename _Operation>
  constexpr void __resize_and_overwrite (size_type __n, _Operation __op)

- template<typename _Operation>
  void __resize_and_overwrite (size_type __n, _Operation __op)
- template<typename _InputIterator>
  basic_string< _CharT, _Traits, _Alloc > & **_M_replace_dispatch** (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2, __false_type)
- template<typename _InIterator>
  _CharT * **_S_construct** (_InIterator __beg, _InIterator __end, const _Alloc &__a, forward_iterator_tag)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr basic_string & append (_InputIterator __first, _InputIterator __last)
- template<class _InputIterator>
  basic_string & append (_InputIterator __first, _InputIterator __last)
- constexpr basic_string & append (const _CharT *__s)
- basic_string & append (const _CharT *__s)
- constexpr basic_string & append (const _CharT *__s, size_type __n)
- basic_string & append (const _CharT *__s, size_type __n)
- constexpr basic_string & append (const basic_string &__str)
- basic_string & append (const basic_string &__str)
- constexpr basic_string & append (const basic_string &__str, size_type __pos, size_type __n=npos)
- basic_string & append (const basic_string &__str, size_type __pos, size_type __n=npos)
- constexpr basic_string & append (initializer_list< _CharT > __l)
- basic_string & append (initializer_list< _CharT > __l)
- constexpr basic_string & append (size_type __n, _CharT __c)
- basic_string & append (size_type __n, _CharT __c)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr basic_string & assign (_InputIterator __first, _InputIterator __last)
- template<class _InputIterator>
  basic_string & assign (_InputIterator __first, _InputIterator __last)
- constexpr basic_string & assign (basic_string &&__str) noexcept(_Alloc_traits::_S_nothrow_move())
- basic_string & assign (basic_string &&__str) noexcept(allocator_traits< _Alloc >::is_always_equal::value)
- constexpr basic_string & assign (const _CharT *__s)
- basic_string & assign (const _CharT *__s)
- constexpr basic_string & assign (const _CharT *__s, size_type __n)
- basic_string & assign (const _CharT *__s, size_type __n)
- constexpr basic_string & assign (const basic_string &__str)
- basic_string & assign (const basic_string &__str)
- constexpr basic_string & assign (const basic_string &__str, size_type __pos, size_type __n=npos)
- basic_string & assign (const basic_string &__str, size_type __pos, size_type __n=npos)
- constexpr basic_string & assign (initializer_list< _CharT > __l)
- basic_string & assign (initializer_list< _CharT > __l)
- constexpr basic_string & assign (size_type __n, _CharT __c)
- basic_string & assign (size_type __n, _CharT __c)
- constexpr reference at (size_type __n)
- reference at (size_type __n)
- constexpr const_reference at (size_type __n) const
- const_reference at (size_type __n) const
- reference back ()
- constexpr const_reference back () const noexcept
- const_reference back () const noexcept
- constexpr reference back () noexcept
- iterator begin ()
- constexpr const_iterator begin () const noexcept

- const_iterator begin () const noexcept
- constexpr iterator begin () noexcept
- constexpr const _CharT * c_str () const noexcept
- const _CharT * c_str () const noexcept
- constexpr size_type capacity () const noexcept
- size_type capacity () const noexcept
- constexpr const_iterator cbegin () const noexcept
- const_iterator cbegin () const noexcept
- constexpr const_iterator cend () const noexcept
- const_iterator cend () const noexcept
- constexpr void clear () noexcept
- void clear () noexcept
- constexpr int compare (const _CharT *__s) const noexcept
- int compare (const _CharT *__s) const noexcept
- constexpr int compare (const basic_string &__str) const
- int compare (const basic_string &__str) const
- constexpr int compare (size_type __pos, size_type __n, const basic_string &__str) const
- int compare (size_type __pos, size_type __n, const basic_string &__str) const
- constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s) const
- int compare (size_type __pos, size_type __n1, const _CharT *__s) const
- constexpr int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- int compare (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2) const
- constexpr int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos) const
- int compare (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos) const
- constexpr bool **contains** (_CharT __x) const noexcept
- bool **contains** (_CharT __x) const noexcept
- constexpr bool **contains** (basic_string_view< _CharT, _Traits > __x) const noexcept
- bool **contains** (basic_string_view< _CharT, _Traits > __x) const noexcept
- constexpr bool **contains** (const _CharT *__x) const noexcept
- bool **contains** (const _CharT *__x) const noexcept
- constexpr size_type copy (_CharT *__s, size_type __n, size_type __pos=0) const
- size_type copy (_CharT *__s, size_type __n, size_type __pos=0) const
- constexpr const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crbegin () const noexcept
- constexpr const_reverse_iterator crend () const noexcept
- const_reverse_iterator crend () const noexcept
- constexpr const _CharT * data () const noexcept
- const _CharT * data () const noexcept
- constexpr _CharT * data () noexcept
- _CharT * data () noexcept(false)
- constexpr bool empty () const noexcept
- bool empty () const noexcept
- iterator end ()
- constexpr const_iterator end () const noexcept
- const_iterator end () const noexcept
- constexpr iterator end () noexcept
- constexpr bool **ends_with** (_CharT __x) const noexcept
- bool **ends_with** (_CharT __x) const noexcept
- constexpr bool **ends_with** (basic_string_view< _CharT, _Traits > __x) const noexcept

- bool **ends_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- constexpr bool **ends_with** (const _CharT *__x) const noexcept
- bool **ends_with** (const _CharT *__x) const noexcept
- constexpr iterator erase (__const_iterator __first, __const_iterator __last)
- constexpr iterator erase (__const_iterator __position)
- iterator erase (iterator __first, iterator __last)
- iterator erase (iterator __position)
- constexpr basic_string & erase (size_type __pos=0, size_type __n=npos)
- basic_string & erase (size_type __pos=0, size_type __n=npos)
- constexpr size_type find (_CharT __c, size_type __pos=0) const noexcept
- size_type find (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find (const _CharT *__s, size_type __pos=0) const noexcept
- size_type find (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_not_of (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_first_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_first_not_of (const _CharT *__s, size_type __pos=0) const noexcept
- size_type find_first_not_of (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find_first_not_of (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- size_type find_first_of (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_first_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_first_of (const _CharT *__s, size_type __pos=0) const noexcept
- size_type find_first_of (const _CharT *__s, size_type __pos=0) const noexcept
- constexpr size_type find_first_of (const basic_string &__str, size_type __pos=0) const noexcept
- size_type find_first_of (const basic_string &__str, size_type __pos=0) const noexcept
- constexpr size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept
- size_type find_last_not_of (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_last_not_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_last_not_of (const _CharT *__s, size_type __pos=npos) const noexcept
- size_type find_last_not_of (const _CharT *__s, size_type __pos=npos) const noexcept
- constexpr size_type find_last_not_of (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_not_of (const basic_string &__str, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- size_type find_last_of (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type find_last_of (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type find_last_of (const _CharT *__s, size_type __pos=npos) const noexcept
- size_type find_last_of (const _CharT *__s, size_type __pos=npos) const noexcept
- constexpr size_type find_last_of (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type find_last_of (const basic_string &__str, size_type __pos=npos) const noexcept
- reference front ()
- constexpr const_reference front () const noexcept

- const_reference front () const noexcept
- constexpr reference front () noexcept
- constexpr allocator_type get_allocator () const noexcept
- allocator_type get_allocator () const noexcept
- constexpr iterator insert (__const_iterator __p, _CharT __c)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr iterator insert (const_iterator __p, _InputIterator __beg, _InputIterator __end)
- constexpr iterator insert (const_iterator __p, initializer_list< _CharT > __l)
- constexpr iterator insert (const_iterator __p, size_type __n, _CharT __c)
- iterator insert (iterator __p, _CharT __c)
- template<class _InputIterator>
  void insert (iterator __p, _InputIterator __beg, _InputIterator __end)
- void insert (iterator __p, initializer_list< _CharT > __l)
- void insert (iterator __p, size_type __n, _CharT __c)
- constexpr basic_string & insert (size_type __pos, const _CharT *__s)
- basic_string & insert (size_type __pos, const _CharT *__s)
- constexpr basic_string & insert (size_type __pos, const _CharT *__s, size_type __n)
- basic_string & insert (size_type __pos, const _CharT *__s, size_type __n)
- constexpr basic_string & insert (size_type __pos, size_type __n, _CharT __c)
- basic_string & insert (size_type __pos, size_type __n, _CharT __c)
- constexpr basic_string & insert (size_type __pos1, const basic_string &__str)
- basic_string & insert (size_type __pos1, const basic_string &__str)
- constexpr basic_string & insert (size_type __pos1, const basic_string &__str, size_type __pos2, size_type __n=npos)
- basic_string & insert (size_type __pos1, const basic_string &__str, size_type __pos2, size_type __n=npos)
- constexpr size_type length () const noexcept
- size_type length () const noexcept
- constexpr size_type max_size () const noexcept
- size_type max_size () const noexcept
- constexpr basic_string & operator+= (_CharT __c)
- basic_string & operator+= (_CharT __c)
- constexpr basic_string & operator+= (const _CharT *__s)
- basic_string & operator+= (const _CharT *__s)
- constexpr basic_string & operator+= (const basic_string &__str)
- basic_string & operator+= (const basic_string &__str)
- constexpr basic_string & operator+= (initializer_list< _CharT > __l)
- basic_string & operator+= (initializer_list< _CharT > __l)
- constexpr basic_string & operator= (_CharT __c)
- basic_string & operator= (_CharT __c)
- basic_string & operator= (basic_string &&__str) noexcept(/*conditional */)
- constexpr basic_string & operator= (basic_string &&__str) noexcept(_Alloc_traits::_S_nothrow_move())
- constexpr basic_string & operator= (const _CharT *__s)
- basic_string & operator= (const _CharT *__s)
- constexpr basic_string & operator= (const basic_string &__str)
- basic_string & operator= (const basic_string &__str)
- constexpr basic_string & operator= (initializer_list< _CharT > __l)
- basic_string & operator= (initializer_list< _CharT > __l)
- basic_string & **operator=** (nullptr_t)=delete
- basic_string & **operator=** (nullptr_t)=delete
- constexpr reference operator[ ] (size_type __pos)
- reference operator[ ] (size_type __pos)

- constexpr const_reference operator[ ] (size_type __pos) const noexcept
- const_reference operator[ ] (size_type __pos) const noexcept
- void pop_back ()
- constexpr void pop_back () noexcept
- constexpr void push_back (_CharT __c)
- void push_back (_CharT __c)
- reverse_iterator rbegin ()
- constexpr const_reverse_iterator rbegin () const noexcept
- const_reverse_iterator rbegin () const noexcept
- constexpr reverse_iterator rbegin () noexcept
- reverse_iterator rend ()
- constexpr const_reverse_iterator rend () const noexcept
- const_reverse_iterator rend () const noexcept
- constexpr reverse_iterator rend () noexcept
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, _CharT *__k1, _CharT *__k2)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const _CharT *__k1, const _CharT *__k2)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const _CharT *__s)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const _CharT *__s, size_type __n)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, const basic_string &__str)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, const_iterator __k1, const_iterator __k2)
- constexpr basic_string & **replace** (__const_iterator __i1, __const_iterator __i2, iterator __k1, iterator __k2)
- constexpr basic_string & replace (__const_iterator __i1, __const_iterator __i2, size_type __n, _CharT __c)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr basic_string & replace (const_iterator __i1, const_iterator __i2, _InputIterator __k1, _InputIterator __k2)
- constexpr basic_string & replace (const_iterator __i1, const_iterator __i2, initializer_list< _CharT > __l)
- basic_string & **replace** (iterator __i1, iterator __i2, _CharT *__k1, _CharT *__k2)
- template<class _InputIterator>
  basic_string & replace (iterator __i1, iterator __i2, _InputIterator __k1, _InputIterator __k2)
- basic_string & **replace** (iterator __i1, iterator __i2, const _CharT *__k1, const _CharT *__k2)
- basic_string & replace (iterator __i1, iterator __i2, const _CharT *__s)
- basic_string & replace (iterator __i1, iterator __i2, const _CharT *__s, size_type __n)
- basic_string & replace (iterator __i1, iterator __i2, const basic_string &__str)
- basic_string & **replace** (iterator __i1, iterator __i2, const_iterator __k1, const_iterator __k2)
- basic_string & replace (iterator __i1, iterator __i2, initializer_list< _CharT > __l)
- basic_string & **replace** (iterator __i1, iterator __i2, iterator __k1, iterator __k2)
- basic_string & replace (iterator __i1, iterator __i2, size_type __n, _CharT __c)
- constexpr basic_string & replace (size_type __pos, size_type __n, const basic_string &__str)
- basic_string & replace (size_type __pos, size_type __n, const basic_string &__str)
- constexpr basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s)
- basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s)
- constexpr basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)
- basic_string & replace (size_type __pos, size_type __n1, const _CharT *__s, size_type __n2)
- constexpr basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- basic_string & replace (size_type __pos, size_type __n1, size_type __n2, _CharT __c)
- constexpr basic_string & replace (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos)
- basic_string & replace (size_type __pos1, size_type __n1, const basic_string &__str, size_type __pos2, size_type __n2=npos)
- constexpr void reserve ()

- void reserve ()
- constexpr void reserve (size_type __res_arg)
- void reserve (size_type __res_arg)
- constexpr void resize (size_type __n)
- void resize (size_type __n)
- constexpr void resize (size_type __n, _CharT __c)
- void resize (size_type __n, _CharT __c)
- constexpr size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- size_type rfind (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- size_type rfind (const _CharT *__s, size_type __pos, size_type __n) const noexcept
- constexpr size_type rfind (const _CharT *__s, size_type __pos=npos) const
- size_type rfind (const _CharT *__s, size_type __pos=npos) const noexcept
- constexpr size_type rfind (const basic_string &__str, size_type __pos=npos) const noexcept
- size_type rfind (const basic_string &__str, size_type __pos=npos) const noexcept
- constexpr void shrink_to_fit () noexcept
- void shrink_to_fit () noexcept
- constexpr size_type size () const noexcept
- size_type size () const noexcept
- constexpr bool **starts_with** (_CharT __x) const noexcept
- bool **starts_with** (_CharT __x) const noexcept
- constexpr bool **starts_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- bool **starts_with** (basic_string_view< _CharT, _Traits > __x) const noexcept
- constexpr bool **starts_with** (const _CharT *__x) const noexcept
- bool **starts_with** (const _CharT *__x) const noexcept
- constexpr basic_string **substr** (size_type __pos, size_type __n) &&
- constexpr basic_string **substr** (size_type __pos=0) &&
- constexpr basic_string substr (size_type __pos=0, size_type __n=npos) const
- basic_string substr (size_type __pos=0, size_type __n=npos) const
- constexpr basic_string_view< _CharT, _Traits > subview (size_type __pos=0, size_type __n=npos) const
- constexpr basic_string_view< _CharT, _Traits > subview (size_type __pos=0, size_type __n=npos) const
- constexpr void swap (basic_string &__s) noexcept
- void swap (basic_string &__s) noexcept(/*conditional */)

Static Public Attributes

- static const size_type npos

Protected Types

- typedef const_iterator **__const_iterator**
- typedef iterator **__const_iterator**

Friends

- template<typename, typename, typename>
  class **basic_stringbuf**

6.249.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_string< _CharT, _Traits, _Alloc >

Managing sequences of characters and character-like objects.

Since

C++98

Template Parameters

_CharT	Type of character
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

Meets the requirements of a container, a reversible container, and a sequence. Of the optional sequence requirements, only push_back, at, and array access are supported.

Since

C++98

Template Parameters

_CharT	Type of character
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

Meets the requirements of a container, a reversible container, and a sequence. Of the optional sequence requirements, only push_back, at, and array access are supported.

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

Documentation? What's that? Nathan Myers ncm@cantrip.org.
A string looks like this:

```
                                        [_Rep]
                                        _M_length
[basic_string<char_type>]               _M_capacity
_M_dataplus                             _M_refcount
_M_p ---------------->                  unnamed array of char_type
```

Where the _M_p points to the first character in the string, and you cast it to a pointer-to-_Rep and subtract 1 to get a pointer to the header.
This approach has the enormous advantage that a string object requires only one allocation. All the ugliness is confined within a single pair of inline functions, which each compile to a single *add* instruction: _Rep::_M_data(), and string::_M_rep(); and the allocation function which gets a block of raw bytes and with room enough and constructs a _Rep object at the front.
The reason you want _M_data pointing to the character array and not the _Rep is so that the debugger can see the string contents. (Probably we should add a non-inline member to get the _Rep for the debugger to use, so users can check the actual string length.)
Note that the _Rep object is a POD so that you can have a static *empty string* _Rep object already *constructed* before static constructors have run. The reference-count encoding is chosen so that a 0 indicates one reference, so you never try to destroy the empty-string _Rep object.
All but the last paragraph is considered pretty conventional for a Copy-On-Write C++ string implementation.

6.249.2 Constructor & Destructor Documentation

basic_string() [1/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string ()  [inline], [constexpr], [noexcept]
```

Default constructor creates an empty string.

Referenced by std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::substr(), swap(), and std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > >::swap().

basic_string() [2/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const _Alloc & __a)  [inline], [explicit], [constexpr], [noexcept]
```

Construct an empty string using allocator *a*.

basic_string() [3/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Construct string with copy of value of *__str*.

Parameters

__str	Source string.

basic_string() [4/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            const _Alloc & __a = _Alloc())  [inline], [constexpr]
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__a	Allocator to use.

basic_string() [5/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n)  [inline], [constexpr]
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy.

basic_string() [6/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n,
            const _Alloc & __a)  [inline], [constexpr]
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy.
__a	Allocator to use.

basic_string() [7/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const _CharT * __s,
            size_type __n,
            const _Alloc & __a = _Alloc())  [inline], [constexpr]
```

Construct string initialized by a character array.

Parameters

_↩ _s	Source character array.
_↩ _n	Number of characters to copy.
_↩ _a	Allocator to use (default is default allocator).

NB: *__s* must have at least *__n* characters, '\0' has no special meaning.

basic_string() [8/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
```

```
template<typename = _RequireAllocator<_Alloc>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const _CharT * __s,
            const _Alloc & __a = _Alloc())  [inline], [constexpr]
```

Construct string as copy of a C string.

Parameters

_↩ _s	Source C string.
_↩ _a	Allocator to use (default is default allocator).

basic_string() [9/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename = _RequireAllocator<_Alloc>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            size_type __n,
            _CharT __c,
            const _Alloc & __a = _Alloc())  [inline], [constexpr]
```

Construct string as multiple characters.

Parameters

_↩ _n	Number of characters.
_↩ _c	Character to use.
_↩ _a	Allocator to use (default is default allocator).

basic_string() [10/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [constexpr], [noexcept]
```

Move construct string.

Parameters

__str	Source string.

The newly-created string contains the exact contents of __str. __str is a valid, but unspecified string.

basic_string() [11/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
```

```
            initializer_list< _CharT > __l,
            const _Alloc & __a = _Alloc())  [inline], [constexpr]
```

Construct string from an initializer list.

Parameters

_↩ _l	std::initializer_list of characters.
_↩ _a	Allocator to use (default is default allocator).

basic_string() [12/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            _InputIterator __beg,
            _InputIterator __end,
            const _Alloc & __a = _Alloc())  [inline], [constexpr]
```

Construct string as copy of a range.

Parameters

__beg	Start of range.
__end	End of range.
__a	Allocator to use (default is default allocator).

~basic_string() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::~basic_string ()  [inline], [constexpr]
```

Destroy the string instance.

basic_string() [13/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string ()  [inline], [noexcept]
```

Default constructor creates an empty string.

basic_string() [14/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const _Alloc & __a)  [inline], [explicit]
```

Construct an empty string using allocator *a*.

basic_string() [15/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Construct string with copy of value of *str*.

Parameters

__str	Source string.

basic_string() [16/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            const _Alloc & __a = _Alloc())
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__a	Allocator to use.

basic_string() [17/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n)
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy.

basic_string() [18/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
```

```
            size_type __n,
            const _Alloc & __a)
```

Construct string as copy of a substring.

Parameters

__str	Source string.
__pos	Index of first character to copy from.
__n	Number of characters to copy.
__a	Allocator to use.

basic_string() [19/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const _CharT * __s,
            size_type __n,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string initialized by a character array.

Parameters

_↩ _s	Source character array.
_↩ _n	Number of characters to copy.
_↩ _a	Allocator to use (default is default allocator).

NB: *__s* must have at least *__n* characters, '\0' has no special meaning.

basic_string() [20/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename = _RequireAllocator<_Alloc>>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            const _CharT * __s,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string as copy of a C string.

Parameters

_↩ _s	Source C string.
_↩ _a	Allocator to use (default is default allocator).

basic_string() [21/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            size_type __n,
            _CharT __c,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string as multiple characters.

Parameters

_↩ _n	Number of characters.
_↩ _c	Character to use.
_↩ _a	Allocator to use (default is default allocator).

basic_string() [22/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [noexcept]
```

Move construct string.

Parameters

__str	Source string.

The newly-created string contains the exact contents of __str. __str is a valid, but unspecified string.

basic_string() [23/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            initializer_list< _CharT > __l,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string from an initializer list.

Parameters

_↩ _l	std::initializer_list of characters.
_↩ _a	Allocator to use (default is default allocator).

basic_string() [24/24]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
std::basic_string< _CharT, _Traits, _Alloc >::basic_string (
            _InputIterator __beg,
            _InputIterator __end,
            const _Alloc & __a = _Alloc())  [inline]
```

Construct string as copy of a range.

Parameters

__beg	Start of range.
__end	End of range.
__a	Allocator to use (default is default allocator).

~basic_string() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_string< _CharT, _Traits, _Alloc >::~basic_string ()  [inline], [noexcept]
```

Destroy the string instance.

6.249.3 Member Function Documentation

__resize_and_overwrite() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string< _CharT, _Traits, _Alloc >::__resize_and_overwrite (
            size_type __n,
            _Operation __op)  [constexpr]
```

Non-standard version of resize_and_overwrite for C++11 and above.

__resize_and_overwrite() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<typename _Operation>
void std::basic_string< _CharT, _Traits, _Alloc >::__resize_and_overwrite (
            size_type __n,
            _Operation __op)
```

Non-standard version of resize_and_overwrite for C++11 and above.

append() [1/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            _InputIterator __first,
            _InputIterator __last)  [inline], [constexpr]
```

Append a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [__first,__last) to this string.

append() [2/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Append a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Appends characters in the range [__first,__last) to this string.

append() [3/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const _CharT * __s)  [inline], [constexpr]
```

Append a C string.

Parameters

_↩ _s	The C string to append.

Returns

Reference to this string.

append() [4/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const _CharT * __s)  [inline]
```

Append a C string.

Parameters

_↩ _s	The C string to append.

Returns

Reference to this string.

append() [5/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const _CharT * __s,
            size_type __n)  [inline], [constexpr]
```

Append a C substring.

Parameters

_↩ _s	The C string to append.
_↩ _n	The number of characters to append.

Returns

Reference to this string.

append() [6/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const _CharT * __s,
            size_type __n)
```

Append a C substring.

Parameters

_↩ _s	The C string to append.
_↩ _n	The number of characters to append.

Returns

Reference to this string.

append() [7/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

Referenced by std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::__cxx11::collate< _CharT >::do_transform(), std::basic_string< _CharT >::operator+=(), std::basic_string< _CharT >::operator+=(), std::basic_string< _CharT >::operator+=(), std::operator>>(), std::basic_string< _CharT, st and resize().

append() [8/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const basic_string< _CharT, _Traits, _Alloc > & __str)
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

append() [9/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n = npos)  [inline], [constexpr]
```

Append a substring.

Parameters

__str	The string to append.
__pos	Index of the first character of str to append.
__n	The number of characters to append.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *__pos* is not a valid index.

This function appends *__n* characters from *__str* starting at *__pos* to this string. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is appended.

append() [10/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n = npos)
```

Append a substring.

Parameters

__str	The string to append.
__pos	Index of the first character of str to append.
__n	The number of characters to append.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *__pos* is not a valid index.

This function appends *__n* characters from *__str* starting at *__pos* to this string. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is appended.

append() [11/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            initializer_list< _CharT > __l)  [inline], [constexpr]
```

Append an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to append.

Returns

Reference to this string.

append() [12/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            initializer_list< _CharT > __l)  [inline]
```

Append an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to append.

Returns

Reference to this string.

append() [13/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::append (
            size_type __n,
            _CharT __c)  [inline], [constexpr]
```

Append multiple characters.

Parameters

_↩ *_n*	The number of characters to append.
_↩ *_c*	The character to use.

Returns

Reference to this string.

Appends __n copies of __c to this string.

append() [14/14]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::append (
            size_type __n,
            _CharT __c)
```

Append multiple characters.

Parameters

_↩ _n	The number of characters to append.
_↩ _c	The character to use.

Returns

Reference to this string.

Appends __n copies of __c to this string.

assign() [1/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline], [constexpr]
```

Set value to a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range [__first,__last).

assign() [2/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Set value to a range of characters.

Parameters

__first	Iterator referencing the first character to append.
__last	Iterator marking the end of the range.

Returns

Reference to this string.

Sets value of string to characters in the range [__first,__last).

assign() [3/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [constexpr], [noexcept]
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of *__str*. *__str* is a valid, but unspecified string.

assign() [4/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [noexcept]
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

This function sets this string to the exact contents of *__str*. *__str* is a valid, but unspecified string.

assign() [5/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const _CharT * __s)  [inline], [constexpr]
```

Set value to contents of a C string.

Parameters

_↩ _s	The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of __s. The data is copied, so there is no dependence on __s once the function returns.

assign() [6/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const _CharT * __s)  [inline]
```

Set value to contents of a C string.

Parameters

_↩ _s	The C string to use.

Returns

Reference to this string.

This function sets the value of this string to the value of __s. The data is copied, so there is no dependence on __s once the function returns.

assign() [7/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const _CharT * __s,
            size_type __n)  [inline], [constexpr]
```

Set value to a C substring.

Parameters

_↩ _s	The C string to use.
_↩ _n	Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first __n characters of __s. If __n is is larger than the number of available characters in __s, the remainder of __s is used.

assign() [8/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const _CharT * __s,
            size_type __n)
```

Set value to a C substring.

Parameters

_↩ _s	The C string to use.
_↩ _n	Number of characters to use.

Returns

Reference to this string.

This function sets the value of this string to the first __n characters of __s. If __n is is larger than the number of available characters in __s, the remainder of __s is used.

assign() [9/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

Referenced by std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::operator=(), std::basic_string< _CharT >::operator=(), std::basic_string< _CharT >::operator=(), std::basic_string< _CharT >::operator=(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< char >::str() and std::basic_stringbuf< char >::str().

assign() [10/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const basic_string< _CharT, _Traits, _Alloc > & __str)
```

Set value to contents of another string.

Parameters

__str	Source string to use.

Returns

Reference to this string.

assign() [11/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n = npos)  [inline], [constexpr]
```

Set value to a substring of a string.

Parameters

__str	The string to use.
__pos	Index of the first character of str.
__n	Number of characters to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *pos* is not a valid index.

This function sets this string to the substring of *__str* consisting of *__n* characters at *__pos*. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is used.

assign() [12/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos,
            size_type __n = npos)  [inline]
```

Set value to a substring of a string.

Parameters

__str	The string to use.
__pos	Index of the first character of str.
__n	Number of characters to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	if *pos* is not a valid index.

This function sets this string to the substring of *__str* consisting of *__n* characters at *__pos*. If *__n* is is larger than the number of available characters in *__str*, the remainder of *__str* is used.

assign() [13/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            initializer_list< _CharT > __l)  [inline], [constexpr]
```

Set value to an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to assign.

Returns

Reference to this string.

assign() [14/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            initializer_list< _CharT > __l)  [inline]
```

Set value to an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to assign.

Returns

Reference to this string.

assign() [15/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
```

```
            size_type __n,
            _CharT __c)  [inline], [constexpr]
```

Set value to multiple characters.

Parameters

_↩ _n	Length of the resulting string.
_↩ _c	The character to use.

Returns

Reference to this string.

This function sets the value of this string to __n copies of character __c.

assign() [16/16]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::assign (
            size_type __n,
            _CharT __c)  [inline]
```

Set value to multiple characters.

Parameters

_↩ _n	Length of the resulting string.
_↩ _c	The character to use.

Returns

Reference to this string.

This function sets the value of this string to __n copies of character __c.

at() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::at (
            size_type __n)  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

at() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::at (
            size_type __n)  [inline]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read/write reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails. Success results in unsharing the string.

at() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::at (
            size_type __n) const  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the string.

Parameters

_↩ _n	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

at() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::at (
            size_type __n) const  [inline]
```

Provides access to the data contained in the string.

Parameters

_↩ _*n*	The index of the character to access.

Returns

Read-only (const) reference to the character.

Exceptions

std::out_of_range	If *n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the string. The function throws out_of_range if the check fails.

back() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::back ()  [inline]
```

Returns a read/write reference to the data at the last element of the string.

back() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::back () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

back() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::back () const  [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the string.

back() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::back () [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write reference to the data at the last element of the string.

begin() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () [inline]
```

Returns a read/write iterator that points to the first character in the string. Unshares the string.

begin() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

begin() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () const [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

begin() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::begin () [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write iterator that points to the first character in the string.
Referenced by std::basic_string< _CharT >::crend(), std::basic_string< _CharT >::erase(), std::basic_string< _CharT >::erase(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::regex_match(), std::regex_match(), std::regex_replace(), std::regex_replace(), std::regex_search(), std::basic_string< _CharT >::ren std::basic_string< _CharT >::rend(), std::basic_string< _CharT >::rend(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), and std::basic_string< _CharT >::replace().

c_str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string< _CharT, _Traits, _Alloc >::c_str () const [inline], [nodiscard],
[constexpr], [noexcept]
```

Return const pointer to null-terminated contents.
This is a handle to internal data. Do not modify or dire things may happen.
Referenced by std::__cxx11::collate< _CharT >::do_compare(), std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::__cxx11::collate< _CharT >::do_transform(), std::basic_filebuf< char_type, traits_type >::open(), std::operator+(), std::operator+(), std::operator+(), std::operator+(), std::operator+(), std::regex_replace(), and std::regex_traits< _CharT >::transform_prim

c_str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string< _CharT, _Traits, _Alloc >::c_str () const [inline], [noexcept]
```

Return const pointer to null-terminated contents.
This is a handle to internal data. Do not modify or dire things may happen.

capacity() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.
Referenced by std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::push_back(), reserve(), and reserve().

capacity() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::capacity () const  [inline], [noexcept]
```

Returns the total number of characters that the string can hold before needing to allocate more memory.

cbegin() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

cbegin() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first character in the string.

cend() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

cend() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

clear() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::clear ()  [inline], [constexpr], [noexcept]
```

Erases the string, making it empty.
Referenced by std::__cxx11::collate< _CharT >::do_transform(), and std::basic_string< _CharT >::operator=().

clear() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::clear ()  [inline], [noexcept]
```

Erases the string, making it empty.

compare() [1/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            const _CharT * __s) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Compare to a C string.

Parameters

_↩ _s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before __s, 0 if their values are equivalent, or > 0 if this string is ordered after __s. Determines the effective length rlen of the strings to compare as the smallest of size() and the length of a string constructed from __s. The function then compares the two strings by calling traits::compare(data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [2/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            const _CharT * __s) const  [inline], [noexcept]
```

Compare to a C string.

Parameters

_↩ _s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before __s, 0 if their values are equivalent, or > 0 if this string is ordered after __s. Determines the effective length rlen of the strings to compare as the smallest of size() and the length of a string constructed from __s. The function then compares the two strings by calling traits::compare(data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [3/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            const basic_string< _CharT, _Traits, _Alloc > & __str) const  [inline], [nodiscard],
[constexpr]
```

Compare to a string.

Parameters

__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *__str*, 0 if their values are equivalent, or > 0 if this string is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of size() and str.size(). The function then compares the two strings by calling traits::compare(data(), str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [4/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            const basic_string< _CharT, _Traits, _Alloc > & __str) const  [inline]
```

Compare to a string.

Parameters

__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Returns an integer < 0 if this string is ordered before *__str*, 0 if their values are equivalent, or > 0 if this string is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of size() and str.size(). The function then compares the two strings by calling traits::compare(data(), str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [5/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Alloc > & __str) const  [inline], [nodiscard],
[constexpr]
```

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [6/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Alloc > & __str) const  [inline]
```

Compare substring to a string.

Parameters

__pos	Index of first character of substring.
__n	Number of characters in substring.
__str	String to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n* characters starting at *__pos*. Returns an integer < 0 if the substring is ordered before *__str*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__str*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__str.size()*. The function then compares the two strings by calling traits::compare(substring.data(),str.data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [7/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s) const  [inline], [nodiscard], [constexpr]
```

Compare substring to a C string.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	C string to compare against.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *pos*. Returns an integer < 0 if the substring is ordered before *__s*, 0 if their values are equivalent, or > 0 if the substring is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *__s*. The function then compares the two string by calling traits::compare(substring.data(),__s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [8/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s) const  [inline]
```

Compare substring to a C string.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	C string to compare against.

Returns

Integer $<$ 0, 0, or $>$ 0.

Form the substring of this string from the *__n1* characters starting at *pos*. Returns an integer $<$ 0 if the substring is ordered before *__s*, 0 if their values are equivalent, or $>$ 0 if the substring is ordered after *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and the length of a string constructed from *__s*. The function then compares the two string by calling traits::compare(substring.data(),__s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [9/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2) const  [inline], [nodiscard], [constexpr]
```

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer $<$ 0, 0, or $>$ 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Form a string from the first *__n2* characters of *__s*. Returns an integer $<$ 0 if this substring is ordered before the string from *__s*, 0 if their values are equivalent, or $>$ 0 if this substring is ordered after the string from *__s*. Determines the effective length rlen of the strings to compare

as the smallest of the length of the substring and __*n2*. The function then compares the two strings by calling traits::↩ compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
NB: s must have at least n2 characters, '\0' has no special meaning.

compare() [10/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2) const  [inline]
```

Compare substring against a character array.

Parameters

__pos	Index of first character of substring.
__n1	Number of characters in substring.
__s	character array to compare against.
__n2	Number of characters of s.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos*. Form a string from the first *__n2* characters of *__s*. Returns an integer < 0 if this substring is ordered before the string from *__s*, 0 if their values are equivalent, or > 0 if this substring is ordered after the string from *__s*. Determines the effective length rlen of the strings to compare as the smallest of the length of the substring and *__n2*. The function then compares the two strings by calling traits::↩ compare(substring.data(),s,rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.
NB: s must have at least n2 characters, '\0' has no special meaning.

compare() [11/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n2 = npos) const  [inline], [nodiscard], [constexpr]
```

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.

__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer < 0 if this substring is ordered before the substring of *__str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *__str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

compare() [12/12]

```
template<typename _CharT, typename _Traits, typename _Alloc>
int std::basic_string< _CharT, _Traits, _Alloc >::compare (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n2 = npos) const  [inline]
```

Compare substring to a substring.

Parameters

__pos1	Index of first character of substring.
__n1	Number of characters in substring.
__str	String to compare against.
__pos2	Index of first character of substring of str.
__n2	Number of characters in substring of str.

Returns

Integer < 0, 0, or > 0.

Form the substring of this string from the *__n1* characters starting at *__pos1*. Form the substring of *__str* from the *__n2* characters starting at *__pos2*. Returns an integer < 0 if this substring is ordered before the substring of *__str*, 0 if their values are equivalent, or > 0 if this substring is ordered after the substring of *__str*. Determines the effective length rlen of the strings to compare as the smallest of the lengths of the substrings. The function then compares the two strings by calling traits::compare(substring.data(),str.substr(pos2,n2).data(),rlen). If the result of the comparison is nonzero returns it, otherwise the shorter one is ordered first.

copy() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::copy (
            _CharT * __s,
            size_type __n,
            size_type __pos = 0) const  [constexpr]
```

Copy substring into C string.

Parameters

__s	C string to copy value into.
__n	Number of characters to copy.
__pos	Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

std::out_of_range	If __pos > size().

Copies up to *__n* characters starting at *__pos* into the C string *__s*. If *__pos* is greater than size(), out_of_range is thrown.

copy() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::copy (
            _CharT * __s,
            size_type __n,
            size_type __pos = 0) const
```

Copy substring into C string.

Parameters

__s	C string to copy value into.
__n	Number of characters to copy.
__pos	Index of first character to copy.

Returns

Number of characters actually copied

Exceptions

std::out_of_range	If __pos > size().

Copies up to *__n* characters starting at *__pos* into the C string *__s*. If *__pos* is greater than size(), out_of_range is thrown.

crbegin() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crbegin () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

crbegin() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

crend() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crend () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

crend() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::crend () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

data() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string< _CharT, _Traits, _Alloc >::data () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Return const pointer to contents.

This is a pointer to internal data. It is undefined to modify the contents through the returned pointer. To get a pointer that allows modifying the contents use `&str[0]` instead, (or in C++17 the non-const `str.data()` overload).

Referenced by std::basic_regex< char >::assign(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::compare(), std::__cxx11::collate< _CharT >::do_compare(), std::__cxx11::collate< _CharT >::do_transfo std::basic_string< _CharT >::find(), std::basic_string< _CharT >::find_first_not_of(), std::basic_string< _CharT >::find_first_of(), std::basic_string< _CharT >::find_last_of(), std::match_results< _BidirectionalIterator, polymorphic_allocator< sub↩ _match< _BidirectionalIterator > > >< const char * >::format(), std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::fr std::operator==(), std::basic_stringbuf< char >::str(), std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes(), and std::regex_traits< _CharT >::transform().

data() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const _CharT * std::basic_string< _CharT, _Traits, _Alloc >::data () const  [inline], [noexcept]
```

Return const pointer to contents.

This is a pointer to internal data. It is undefined to modify the contents through the returned pointer. To get a pointer that allows modifying the contents use `&str[0]` instead, (or in C++17 the non-const `str.data()` overload).

data() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
_CharT * std::basic_string< _CharT, _Traits, _Alloc >::data ()  [inline], [nodiscard], [constexpr],
[noexcept]
```

Return non-const pointer to contents.
This is a pointer to the character sequence held by the string. Modifying the characters in the sequence is allowed.

data() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
_CharT * std::basic_string< _CharT, _Traits, _Alloc >::data ()  [inline]
```

Return non-const pointer to contents.
This is a pointer to the character sequence held by the string. Modifying the characters in the sequence is allowed.
The standard requires this function to be `noexcept` but for the Copy-On-Write string implementation it can throw. This function allows modifying the string contents directly, which means we must copy-on-write to unshare it, which requires allocating memory.

empty() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::basic_string< _CharT, _Traits, _Alloc >::empty () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.
Referenced by std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::front(), std::basic_string< _CharT >::front(), std::basic_string< _CharT >::front(), std::regex_traits< _Ch_type >::lookup_collatename(), std::tr2::operator>>(), std::basic_string< _CharT >::pop_back(), and std::basic_string< _CharT >::pop_back().

empty() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
bool std::basic_string< _CharT, _Traits, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the string is empty. Equivalent to `*this == ""`.

end() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::end ()  [inline]
```

Returns a read/write iterator that points one past the last character in the string. Unshares the string.

end() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::end () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

end() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_iterator std::basic_string< _CharT, _Traits, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last character in the string.

end() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::end ()  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write iterator that points one past the last character in the string.

Referenced by std::basic_string< _CharT >::append(), std::basic_string< _CharT >::crbegin(), std::basic_string< _CharT >::erase(), std::basic_string< _CharT >::erase(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::rbegin(), std::basic_string< _CharT >::rbegin(), std::basic_string< _CharT >::rbegin(), std::regex_match(), std::regex_match(), std::regex_replace(), std::regex_replace(), std::regex_search(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), and std::basic_string< _CharT >::r

erase() [1/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
            __const_iterator __first,
            __const_iterator __last)  [inline], [constexpr]
```

Remove a range of characters.

Parameters

__first	Iterator referencing the first character to remove.
__last	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

erase() [2/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
            __const_iterator __position)  [inline], [constexpr]
```

Remove one character.

Parameters

__position	Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *__position* from this string. The value of the string doesn't change if an error is thrown.

erase() [3/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::iterator std::basic_string< _CharT, _Traits, _Alloc
>::erase (
```

```
            iterator __first,
            iterator __last)
```

Remove a range of characters.

Parameters

__first	Iterator referencing the first character to remove.
__last	Iterator referencing the end of the range.

Returns

Iterator referencing location of first after removal.

Removes the characters in the range [first,last) from this string. The value of the string doesn't change if an error is thrown.

erase() [4/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::erase (
            iterator __position)  [inline]
```

Remove one character.

Parameters

__position	Iterator referencing the character to remove.

Returns

iterator referencing same location after removal.

Removes the character at *__position* from this string. The value of the string doesn't change if an error is thrown.

erase() [5/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::erase (
            size_type __pos = 0,
            size_type __n = npos)  [inline], [constexpr]
```

Remove characters.

Parameters

__pos	Index of first character to remove (default 0).
__n	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.

Removes *__n* characters from this string starting at *__pos*. The length of the string is reduced by *__n*. If there are < *__n* characters to remove, the remainder of the string is truncated. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::getline(), std::operator>>(), std::basic_string< _CharT >::pop_back(), and resize().

erase() [6/6]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::erase (
            size_type __pos = 0,
            size_type __n = npos)  [inline]
```

Remove characters.

Parameters

__pos	Index of first character to remove (default 0).
__n	Number of characters to remove (default remainder).

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.

Removes *__n* characters from this string starting at *__pos*. The length of the string is reduced by *__n*. If there are < *__n* characters to remove, the remainder of the string is truncated. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

find() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
            _CharT __c,
            size_type __pos = 0) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.

__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            _CharT __c,
            size_type __pos = 0) const  [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from __pos, searches forward for the first __n characters in __s within this string. If found, returns the index where it begins. If not found, returns npos.
References std::data(), npos, and std::size().
Referenced by std::basic_string< _CharT >::find(), std::basic_string< _CharT >::find(), std::basic_string< _CharT >::find(), and std::basic_string< _CharT >::find_first_of().

find() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search from.
__n	Number of characters from *s* to search for.

Returns

Index of start of first occurrence.

Starting from __pos, searches forward for the first __n characters in __s within this string. If found, returns the index where it begins. If not found, returns npos.

find() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from __pos, searches forward for the value of __s within this string. If found, returns the index where it begins. If not found, returns npos.

find() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

find() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a string.

Parameters

__str	String to locate.

__pos	Index of character to search from (default 0).

Returns

Index of start of first occurrence.

Starting from *__pos*, searches forward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

find_first_not_of() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
            _CharT __c,
            size_type __pos = 0) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_first_not_of() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            _CharT __c,
            size_type __pos = 0) const  [noexcept]
```

Find position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from __s to consider.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_first_not_of() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from.
__n	Number of characters from __s to consider.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
```

```
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_not_of() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_first_not_of(), std::basic_string< _CharT >::find_first_not_of(), and std::basic_string< _CharT >::find_first_not_of().

find_first_not_of() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_not_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            _CharT __c,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to find(__c, __pos).

find_first_of() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
```

```
            _CharT __c,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for the character *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to find(__c, __pos).

find_first_of() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_first_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_first_of() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find position of a character of C substring.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from.
__n	Number of characters from s to search for.

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of C string.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const _CharT * __s,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character of C string.

Parameters

__s	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_first_of() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_first_of(), and std::basic_string< _CharT >::find_first_of().

find_first_of() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_first_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = 0) const  [inline], [noexcept]
```

Find position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search from (default 0).

Returns

Index of first occurrence.

Starting from *__pos*, searches forward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
```

```
            _CharT __c,
            size_type __pos = npos) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_last_not_of() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            _CharT __c,
            size_type __pos = npos) const  [noexcept]
```

Find last position of a different character.

Parameters

__c	Character to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character other than *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.

__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_last_not_of() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find last position of a character not in C substring.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from.
__n	Number of characters from s to consider.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character not in C string.

Parameters

__s	C string containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_not_of() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_last_not_of(), std::basic_string< _CharT >::find_last_not_of(), and std::basic_string< _CharT >::find_last_not_of().

find_last_not_of() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_not_of (
```

```
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character not in string.

Parameters

__str	String containing characters to avoid.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for a character not contained in *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            _CharT __c,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to rfind(__c, __pos).

find_last_of() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            _CharT __c,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
Note: equivalent to rfind(__c, __pos).

find_last_of() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::find_last_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C substring.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

find_last_of() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find last position of a character of C substring.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the first *__n* characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of C string.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character of C string.

Parameters

__s	C string containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__s* within this string. If found, returns the index where it was found. If not found, returns npos.

find_last_of() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
```

```
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_last_of(), and std::basic_string< _CharT >::find_last_of().

find_last_of() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::find_last_of (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a character of string.

Parameters

__str	String containing characters to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for one of the characters of *__str* within this string. If found, returns the index where it was found. If not found, returns npos.

front() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::front ()  [inline]
```

Returns a read/write reference to the data at the first element of the string.

front() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::front () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

front() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::front () const  [inline], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the string.

front() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::front ()  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read/write reference to the data at the first element of the string.

get_allocator() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
allocator_type std::basic_string< _CharT, _Traits, _Alloc >::get_allocator () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Return copy of allocator used to construct this string.

Referenced by std::basic_string< _CharT >::~basic_string(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::clear
std::operator+(), std::operator+(), std::operator+(), std::operator+(), std::operator+(), reserve(), reserve(), std::basic_string< _CharT, std::c
and swap().

get_allocator() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
allocator_type std::basic_string< _CharT, _Traits, _Alloc >::get_allocator () const  [inline],
[noexcept]
```

Return copy of allocator used to construct this string.

insert() [1/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            __const_iterator __p,
            _CharT __c)  [inline], [constexpr]
```

Insert one character.

Parameters

_↩ _p	Iterator referencing position in string to insert at.
_↩ _c	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts character __c at position referenced by __p. If adding character causes the length to exceed max_size(), length_error is thrown. If __p is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [2/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            const_iterator __p,
            _InputIterator __beg,
            _InputIterator __end)  [inline], [constexpr]
```

Insert a range of characters.

Parameters

__p	Const_iterator referencing location in string to insert at.
__beg	Start of range.
__end	End of range.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts characters in range [beg,end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [3/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            const_iterator __p,
            initializer_list< _CharT > __l)  [inline], [constexpr]
```

Insert an initializer_list of characters.

Parameters

_↩ _p	Iterator referencing location in string to insert at.
_↩ _l	The initializer_list of characters to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

insert() [4/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            const_iterator __p,
            size_type __n,
            _CharT __c)  [inline], [constexpr]
```

Insert multiple characters.

Parameters

_↩ _p	Const_iterator referencing location in string to insert at.
_↩ _n	Number of characters to insert
_↩ _c	The character to insert.

Returns

Iterator referencing the first inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts __n copies of character __c starting at the position referenced by iterator __p. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), and std::basic_string< _CharT >::insert().

insert() [5/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
iterator std::basic_string< _CharT, _Traits, _Alloc >::insert (
            iterator __p,
            _CharT __c)  [inline]
```

Insert one character.

Parameters

_↩ _p	Iterator referencing position in string to insert at.
_↩ _c	The character to insert.

Returns

Iterator referencing newly inserted char.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts character *__c* at position referenced by *__p*. If adding character causes the length to exceed max_size(), length_error is thrown. If *__p* is beyond end of string, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [6/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
             iterator __p,
             _InputIterator __beg,
             _InputIterator __end)  [inline]
```

Insert a range of characters.

Parameters

__p	Iterator referencing location in string to insert at.
__beg	Start of range.
__end	End of range.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts characters in range [__beg,__end). If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [7/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
             iterator __p,
             initializer_list< _CharT > __l)  [inline]
```

Insert an initializer_list of characters.

Parameters

__↩ __p	Iterator referencing location in string to insert at.

_↩ _l	The initializer_list of characters to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

insert() [8/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::insert (
            iterator __p,
            size_type __n,
            _CharT __c)  [inline]
```

Insert multiple characters.

Parameters

_↩ _p	Iterator referencing location in string to insert at.
_↩ _n	Number of characters to insert
_↩ _c	The character to insert.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts *__n* copies of character *__c* starting at the position referenced by iterator *__p*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [9/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            const _CharT * __s)  [inline], [constexpr]
```

Insert a C string.

Parameters

__pos	Position in string to insert at.

__s	The C string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos* is beyond the end of this string.

Inserts the first *n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [10/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            const _CharT * __s)  [inline]
```

Insert a C string.

Parameters

__pos	Position in string to insert at.
__s	The C string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos* is beyond the end of this string.

Inserts the first *n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [11/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            const _CharT * __s,
            size_type __n)  [inline], [constexpr]
```

Insert a C substring.

Parameters

__pos	Position in string to insert at.
__s	The C string to insert.
__n	The number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts the first *__n* characters of *__s* starting at *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [12/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            const _CharT * __s,
            size_type __n)
```

Insert a C substring.

Parameters

__pos	Position in string to insert at.
__s	The C string to insert.
__n	The number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts the first __n characters of __s starting at __pos. If adding characters causes the length to exceed max_size(), length_error is thrown. If __pos is beyond end(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [13/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            size_type __n,
            _CharT __c)  [inline], [constexpr]
```

Insert multiple characters.

Parameters

__pos	Index in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts __n copies of character __c starting at index __pos. If adding characters causes the length to exceed max_size(), length_error is thrown. If __pos > length(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [14/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos,
            size_type __n,
            _CharT __c)  [inline]
```

Insert multiple characters.

Parameters

__pos	Index in string to insert at.
__n	Number of characters to insert
__c	The character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *__pos* is beyond the end of this string.

Inserts *__n* copies of character *__c* starting at index *__pos*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos* > length(), out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [15/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos1,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Insert value of a string.

Parameters

__pos1	Position in string to insert at.
__str	The string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts value of *__str* starting at *__pos1*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [16/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos1,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Insert value of a string.

Parameters

__pos1	Position in string to insert at.
__str	The string to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Inserts value of *__str* starting at *__pos1*. If adding characters causes the length to exceed max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

insert() [17/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n = npos)  [inline], [constexpr]
```

Insert a substring.

Parameters

__pos1	Position in string to insert at.
__str	The string to insert.
__pos2	Start of characters in str to insert.
__n	Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos1* > size() or *__pos2* > *str.size()*.

Starting at *pos1*, insert *__n* character of *__str* beginning with *__pos2*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos1* is beyond the end of this string or *__pos2* is beyond the end of *__str*, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

insert() [18/18]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::insert (
            size_type __pos1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n = npos)  [inline]
```

Insert a substring.

Parameters

__pos1	Position in string to insert at.
__str	The string to insert.
__pos2	Start of characters in str to insert.
__n	Number of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.
std::out_of_range	If *pos1* > size() or *__pos2* > *str.size()*.

Starting at *pos1*, insert *__n* character of *__str* beginning with *__pos2*. If adding characters causes the length to exceed max_size(), length_error is thrown. If *__pos1* is beyond the end of this string or *__pos2* is beyond the end of *__str*, out_of_range is thrown. The value of the string doesn't change if an error is thrown.

length() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::length () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.
Referenced by std::basic_string< _CharT >::basic_string(), std::basic_string< _CharT >::basic_string(), std::__cxx11::collate< _CharT std::__cxx11::collate< _CharT >::do_transform(), std::basic_string< _CharT >::operator=(), std::regex_replace(), and reserve().

length() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::length () const  [inline], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

max_size() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::max_size () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the size() of the largest possible string.
Referenced by std::getline(), and std::operator>>().

max_size() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the size() of the largest possible string.

operator+=() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            _CharT __c)  [inline], [constexpr]
```

Append a character.

Parameters

_↩ _c	The character to append.

Returns

Reference to this string.

operator+=() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            _CharT __c)  [inline]
```

Append a character.

Parameters

_↩ _c	The character to append.

Returns

Reference to this string.

operator+=() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            const _CharT * __s)  [inline], [constexpr]
```

Append a C string.

Parameters

_↩ _s	The C string to append.

Returns

Reference to this string.

operator+=() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            const _CharT * __s)  [inline]
```

Append a C string.

Parameters

_↩ _s	The C string to append.

Returns

Reference to this string.

operator+=() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

operator+=() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Append a string to this string.

Parameters

__str	The string to append.

Returns

Reference to this string.

operator+=() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            initializer_list< _CharT > __l)  [inline], [constexpr]
```

Append an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to be appended.

Returns

Reference to this string.

operator+=() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator+= (
            initializer_list< _CharT > __l)  [inline]
```

Append an initializer_list of characters.

Parameters

↩ _↩ ↩ _↩ *l*	The initializer_list of characters to be appended.

Returns

Reference to this string.

operator=() [1/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            _CharT __c)  [inline], [constexpr]
```

Set value to string of length 1.

Parameters

_↩ *_c*	Source character.

Assigning to a character makes this string length 1 and (∗this)[0] == *c*.

operator=() [2/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            _CharT __c)  [inline]
```

Set value to string of length 1.

Parameters

_↩ _*c*	Source character.

Assigning to a character makes this string length 1 and (∗this)[0] == *c*.

operator=() [3/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [noexcept]
```

Move assign the value of *str* to this string.

Parameters

__*str*	Source string.

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

operator=() [4/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            basic_string< _CharT, _Traits, _Alloc > && __str)  [inline], [constexpr], [noexcept]
```

Move assign the value of *str* to this string.

Parameters

__*str*	Source string.

The contents of *str* are moved into this string (without copying). *str* is a valid, but unspecified string.

operator=() [5/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            const _CharT * __s)  [inline], [constexpr]
```

Copy contents of *s* into this string.

Parameters

_↩ _*s*	Source null-terminated string.

operator=() [6/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            const _CharT * __s)  [inline]
```

Copy contents of *s* into this string.

Parameters

_↩ _s	Source null-terminated string.

operator=() [7/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Assign the value of *str* to this string.

Parameters

__str	Source string.

operator=() [8/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Assign the value of *str* to this string.

Parameters

__str	Source string.

operator=() [9/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            initializer_list< _CharT > __l)  [inline], [constexpr]
```

Set value to string constructed from initializer list.

Parameters

↩ _↩ ↩ _↩ l	std::initializer_list.

operator=() [10/10]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::operator= (
            initializer_list< _CharT > __l)  [inline]
```

Set value to string constructed from initializer list.

Parameters

↩ _↩ ↩ _↩ *l*	std::initializer_list.

operator[]() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
            size_type __pos)  [inline], [nodiscard], [constexpr]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

operator[]() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
            size_type __pos)  [inline]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read/write reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().) Unshares the string.

operator[]() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
            size_type __pos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)
Referenced by std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::front(), std::basic_string< _CharT >::front(), and std::basic_string< _CharT >::front().

operator[]() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reference std::basic_string< _CharT, _Traits, _Alloc >::operator[] (
            size_type __pos) const  [inline], [noexcept]
```

Subscript access to the data contained in the string.

Parameters

__pos	The index of the character to access.

Returns

Read-only (constant) reference to the character.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

pop_back() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::pop_back ()  [inline]
```

Remove the last character.
The string must be non-empty.

pop_back() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::pop_back ()  [inline], [constexpr], [noexcept]
```

Remove the last character.
The string must be non-empty.

push_back() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::push_back (
            _CharT __c)  [inline], [constexpr]
```

Append a single character.

Parameters

_↩ _c	Character to append.

Referenced by std::__cxx11::collate< _CharT >::do_transform(), std::basic_string< _CharT >::operator+=(), std::tr2::operator>>(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

push_back() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::push_back (
            _CharT __c)  [inline]
```

Append a single character.

Parameters

_↩ _c	Character to append.

rbegin() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin ()  [inline]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order. Unshares the string.

rbegin() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

rbegin() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

rbegin() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rbegin ()  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to the last character in the string. Iteration is done in reverse element order.

rend() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend ()  [inline]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order. Unshares the string.

rend() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend () const  [inline],
[nodiscard], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

rend() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
const_reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

rend() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
reverse_iterator std::basic_string< _CharT, _Traits, _Alloc >::rend ()  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first character in the string. Iteration is done in reverse element order.

replace() [1/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            const _CharT * __s)  [inline], [constexpr]
```

Replace range of characters with C string.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _s	C string value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [2/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            const _CharT * __s,
            size_type __n)  [inline], [constexpr]
```

Replace range of characters with C substring.

Parameters

_↩ _*i1*	Iterator referencing start of range to replace.
_↩ _*i2*	Iterator referencing end of range to replace.
_↩ _*s*	C string value to insert.
_↩ _*n*	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the first __*n* characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [3/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Replace range of characters with string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__str	String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the value of *__str* is inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [4/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            __const_iterator __i1,
            __const_iterator __i2,
            size_type __n,
            _CharT __c)  [inline], [constexpr]
```

Replace range of characters with multiple characters.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _n	Number of characters to insert.
_↩ _c	Character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, *__n* copies of *__c* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [5/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            const_iterator __i1,
            const_iterator __i2,
            _InputIterator __k1,
            _InputIterator __k2)  [inline], [constexpr]
```

Replace range of characters with range.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__k1	Iterator referencing start of range to insert.
__k2	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [6/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            const_iterator __i1,
            const_iterator __i2,
            initializer_list< _CharT > __l)  [inline], [constexpr]
```

Replace range of characters with initializer_list.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.

_↩ _l	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [7/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
template<class _InputIterator>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            _InputIterator __k1,
            _InputIterator __k2)  [inline]
```

Replace range of characters with range.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__k1	Iterator referencing start of range to insert.
__k2	Iterator referencing end of range to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [8/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            const _CharT * __s)  [inline]
```

Replace range of characters with C string.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _s	C string value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the characters of __*s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [9/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            const _CharT * __s,
            size_type __n)  [inline]
```

Replace range of characters with C substring.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _s	C string value to insert.
_↩ _n	Number of characters from s to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the first *__n* characters of *__s* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [10/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Replace range of characters with string.

Parameters

__i1	Iterator referencing start of range to replace.
__i2	Iterator referencing end of range to replace.
__str	String value to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, the value of *__str* is inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [11/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            iterator __i1,
            iterator __i2,
            initializer_list< _CharT > __l)  [inline]
```

Replace range of characters with initializer_list.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _l	The initializer_list of characters to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, characters in the range [__k1,__k2) are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [12/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
             iterator __i1,
             iterator __i2,
             size_type __n,
             _CharT __c)  [inline]
```

Replace range of characters with multiple characters.

Parameters

_↩ _i1	Iterator referencing start of range to replace.
_↩ _i2	Iterator referencing end of range to replace.
_↩ _n	Number of characters to insert.
_↩ _c	Character to insert.

Returns

Reference to this string.

Exceptions

std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__i1,__i2). In place, __*n* copies of __*c* are inserted. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [13/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline], [constexpr]
```

Replace characters with value from another string.

Parameters

__pos	Index of first character to replace.
__n	Number of characters to be replaced.
__str	String to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos+__n) from this string. In place, the value of __*str* is inserted. If __*pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.
Referenced by std::basic_string< _CharT >::append(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), and std::basic_string< _CharT >::replace().

replace() [14/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
```

```
            size_type __n,
            const basic_string< _CharT, _Traits, _Alloc > & __str)  [inline]
```

Replace characters with value from another string.

Parameters

__pos	Index of first character to replace.
__n	Number of characters to be replaced.
__str	String to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* is beyond the end of this string.
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos+__n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [15/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n1,
            const _CharT * __s)  [inline], [constexpr]
```

Replace characters with value of a C string.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the characters of __*s* are inserted. If __*pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [16/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n1,
            const _CharT * __s)  [inline]
```

Replace characters with value of a C string.

Parameters

__*pos*	Index of first character to replace.
__*n1*	Number of characters to be replaced.
__*s*	C string to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the characters of __*s* are inserted. If __*pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [17/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc > & std::basic_string< _CharT, _Traits, _Alloc >::replace
(
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2)  [inline], [constexpr]
```

Replace characters with value of a C substring.

Parameters

__*pos*	Index of first character to replace.
__*n1*	Number of characters to be replaced.
__*s*	C string to insert.

__n2	Number of characters from *s* to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos1* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the first *__n2* characters of *__s* are inserted, or all of *__s* if *__n2* is too large. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [18/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n1,
            const _CharT * __s,
            size_type __n2)
```

Replace characters with value of a C substring.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__s	C string to insert.
__n2	Number of characters from *s* to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *pos1* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos,__pos + __n1) from this string. In place, the first *__n2* characters of *__s* are inserted, or all of *__s* if *__n2* is too large. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [19/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
```

```
            size_type __pos,
            size_type __n1,
            size_type __n2,
            _CharT __c)  [inline], [constexpr]
```

Replace characters with multiple characters.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__n2	Number of characters to insert.
__c	Character to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, *__n2* copies of *__c* are inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [20/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos,
            size_type __n1,
            size_type __n2,
            _CharT __c)  [inline]
```

Replace characters with multiple characters.

Parameters

__pos	Index of first character to replace.
__n1	Number of characters to be replaced.
__n2	Number of characters to insert.
__c	Character to insert.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos* > size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [pos,pos + n1) from this string. In place, *__n2* copies of *__c* are inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [21/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos1,
            size_type __n1,
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos2,
            size_type __n2 = npos)  [inline], [constexpr]
```

Replace characters with value from another string.

Parameters

__pos1	Index of first character to replace.
__n1	Number of characters to be replaced.
__str	String to insert.
__pos2	Index of first character of str to use.
__n2	Number of characters from str to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size() or *__pos2* > __str.size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos1,__pos1 + n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

replace() [22/22]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string & std::basic_string< _CharT, _Traits, _Alloc >::replace (
            size_type __pos1,
            size_type __n1,
```

```
        const basic_string< _CharT, _Traits, _Alloc > & __str,
        size_type __pos2,
        size_type __n2 = npos)  [inline]
```

Replace characters with value from another string.

Parameters

__pos1	Index of first character to replace.
__n1	Number of characters to be replaced.
__str	String to insert.
__pos2	Index of first character of str to use.
__n2	Number of characters from str to use.

Returns

Reference to this string.

Exceptions

std::out_of_range	If *__pos1* > size() or *__pos2* > __str.size().
std::length_error	If new length exceeds `max_size()`.

Removes the characters in the range [__pos1,__pos1 + n) from this string. In place, the value of *__str* is inserted. If *__pos* is beyond end of string, out_of_range is thrown. If the length of the result exceeds max_size(), length_error is thrown. The value of the string doesn't change if an error is thrown.

reserve() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve ()  [constexpr]
```

Equivalent to shrink_to_fit().
References capacity(), get_allocator(), and length().
Referenced by std::basic_string< _CharT >::push_back(), and std::basic_string< _CharT >::shrink_to_fit().

reserve() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve ()
```

Equivalent to shrink_to_fit().

reserve() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve (
            size_type __res_arg)  [constexpr]
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.
References capacity(), get_allocator(), and std::size().
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:
std::tr2::operator>>(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow().

reserve() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::reserve (
            size_type __res_arg)
```

Attempt to preallocate enough memory for specified number of characters.

Parameters

__res_arg	Number of characters required.

Exceptions

std::length_error	If *__res_arg* exceeds `max_size()`.

This function attempts to reserve enough memory for the string to hold the specified number of characters. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the string length that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of string data.

resize() [1/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
            size_type __n)  [inline], [constexpr]
```

Resizes the string to the specified number of characters.

Parameters

__n	Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

resize() [2/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
            size_type __n)  [inline]
```

Resizes the string to the specified number of characters.

Parameters

_↩ _n	Number of characters the string should contain.

This function will resize the string to the specified length. If the new size is smaller than the string's current size the string is truncated, otherwise the string is extended and new characters are default-constructed. For basic types such as char, this means setting them to 0.

resize() [3/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
            size_type __n,
            _CharT __c)  [constexpr]
```

Resizes the string to the specified number of characters.

Parameters

_↩ _n	Number of characters the string should contain.
_↩ _c	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to __c.
References append(), erase(), and size().
Referenced by std::basic_string< _CharT >::resize().

resize() [4/4]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::resize (
            size_type __n,
            _CharT __c)
```

Resizes the string to the specified number of characters.

Parameters

_↩ _n	Number of characters the string should contain.
_↩ _c	Character to fill any new elements.

This function will resize the string to the specified number of characters. If the number is smaller than the string's current size the string is truncated, otherwise the string is extended and new elements are set to __c.

rfind() [1/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
            _CharT __c,
            size_type __pos = npos) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.
References npos, and std::size().

rfind() [2/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            _CharT __c,
            size_type __pos = npos) const  [noexcept]
```

Find last position of a character.

Parameters

__c	Character to locate.
__pos	Index of character to search back from (default end).

Returns

Index of last occurrence.

Starting from *__pos*, searches backward for *__c* within this string. If found, returns the index where it was found. If not found, returns npos.

rfind() [3/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string< _CharT, _Traits, _Alloc >::size_type std::basic_string< _CharT, _Traits, _Alloc
>::rfind (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [nodiscard], [constexpr], [noexcept]
```

Find last position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.
References std::min(), npos, and std::size().

rfind() [4/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const _CharT * __s,
            size_type __pos,
            size_type __n) const  [noexcept]
```

Find last position of a C substring.

Parameters

__s	C string to locate.
__pos	Index of character to search back from.
__n	Number of characters from s to search for.

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the first *__n* characters in *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [5/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr]
```

Find last position of a C string.

Parameters

__s	C string to locate.

__pos	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [6/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const _CharT * __s,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a C string.

Parameters

__s	C string to locate.
__pos	Index of character to start search at (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for the value of *__s* within this string. If found, returns the index where it begins. If not found, returns npos.

rfind() [7/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.
Referenced by std::basic_string< _CharT >::find_last_of(), std::basic_string< _CharT >::rfind(), std::basic_string< _CharT >::rfind(), std::basic_string< _CharT >::rfind(), and std::basic_string< _CharT >::rfind().

rfind() [8/8]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::rfind (
            const basic_string< _CharT, _Traits, _Alloc > & __str,
            size_type __pos = npos) const  [inline], [noexcept]
```

Find last position of a string.

Parameters

__str	String to locate.
__pos	Index of character to search back from (default end).

Returns

Index of start of last occurrence.

Starting from *__pos*, searches backward for value of *__str* within this string. If found, returns the index where it begins. If not found, returns npos.

shrink_to_fit() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::shrink_to_fit ()  [inline], [constexpr], [noexcept]
```

A non-binding request to reduce capacity() to size().

shrink_to_fit() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::shrink_to_fit ()  [inline], [noexcept]
```

A non-binding request to reduce capacity() to size().

size() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::size () const  [inline], [nodiscard],
[constexpr], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

Referenced by std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset(), std::basic_string< _CharT >::append(), std::basic_string< _CharT >::append(), std::basic_regex< char >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::assign(), std::basic_string< _CharT >::at(), std::basic_string< _CharT >::at(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::back(), std::basic_string< _CharT >::cend(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::compare(), std::basic_string< _CharT >::empty(), std::basic_string< _CharT >::end(), std::basic_string< _CharT >::end(), std::basic_string< _CharT >::end(), std::basic_string< _CharT >::find(), std::basic_string< _CharT >::find_first_not_of(), std::basic_string< _CharT >::find_first_of(), std::basic_string< _CharT >::find_last_of(), std::match_results< _BidirectionalIterator, polymorphic_allocator< sub↩ _match< _BidirectionalIterator > > >< const char * >::format(), std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::fr std::basic_string< _CharT >::insert(), std::basic_string< _CharT >::length(), std::operator+(), std::operator+(), std::operator+(), std::operator+(), std::operator+(), std::basic_string< _CharT >::operator=(), std::operator==(), std::tr2::operator>>(), std::basic_string< _CharT >::operator[](), std::basic_string< _CharT >::operator[](), std::basic_string< _CharT std::basic_string< _CharT >::pop_back(), std::basic_string< _CharT >::push_back(), std::basic_string< _CharT >::replace(), std::basic_string< _CharT >::replace(), resize(), std::basic_stringbuf< char >::str(), std::wstring_convert< _Codecvt, _Elem, _Wide_allo std::regex_traits< _CharT >::transform(), and std::regex_traits< _CharT >::transform_primary().

size() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
size_type std::basic_string< _CharT, _Traits, _Alloc >::size () const  [inline], [noexcept]
```

Returns the number of characters in the string, not including any null-termination.

substr() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (
            size_type __pos = 0,
            size_type __n = npos) const  [inline], [nodiscard], [constexpr]
```

Get a substring.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

std::out_of_range	If __pos > size().

Construct and return a new string using the *__n* characters starting at *__pos*. If the string is too short, use the remainder of the characters. If *__pos* is beyond the end of the string, out_of_range is thrown.

substr() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string std::basic_string< _CharT, _Traits, _Alloc >::substr (
            size_type __pos = 0,
            size_type __n = npos) const  [inline]
```

Get a substring.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in substring (default remainder).

Returns

The new string.

Exceptions

std::out_of_range	If __pos > size().

Construct and return a new string using the *__n* characters starting at *__pos*. If the string is too short, use the remainder of the characters. If *__pos* is beyond the end of the string, out_of_range is thrown.

subview() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string_view< _CharT, _Traits > std::basic_string< _CharT, _Traits, _Alloc >::subview (
            size_type __pos = 0,
            size_type __n = npos) const  [inline], [nodiscard], [constexpr]
```

Get a subview.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in subview (default remainder).

Returns

The subview.

Exceptions

std::out_of_range	If __pos > size().

Construct and return a subview using the __n characters starting at __pos. If the string is too short, use the remainder of the characters. If __pos is beyond the end of the string, out_of_range is thrown.

subview() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
basic_string_view< _CharT, _Traits > std::basic_string< _CharT, _Traits, _Alloc >::subview (
            size_type __pos = 0,
            size_type __n = npos) const  [inline], [nodiscard], [constexpr]
```

Get a subview.

Parameters

__pos	Index of first character (default 0).
__n	Number of characters in subview (default remainder).

Returns

The subview.

Exceptions

std::out_of_range	If __pos > size().

Construct and return a subview using the __n characters starting at __pos. If the string is too short, use the remainder of the characters. If __pos is beyond the end of the string, out_of_range is thrown.

swap() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::swap (
            basic_string< _CharT, _Traits, _Alloc > & __s)  [constexpr], [noexcept]
```

Swap contents with another string.

Parameters

_↩ _s	String to swap with.

Exchanges the contents of this string with that of __s in constant time.
References basic_string(), and get_allocator().
Referenced by std::basic_string< _CharT >::assign(), and std::basic_string< _CharT >::operator=().

swap() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_string< _CharT, _Traits, _Alloc >::swap (
            basic_string< _CharT, _Traits, _Alloc > & __s)  [noexcept]
```

Swap contents with another string.

Parameters

_↩ _s	String to swap with.

Exchanges the contents of this string with that of __s in constant time.

6.249.4 Member Data Documentation

npos

```
template<typename _CharT, typename _Traits, typename _Alloc>
const size_type std::basic_string< _CharT, _Traits, _Alloc >::npos  [static]
```

Value returned by various member functions when they fail.
Referenced by std::bitset< _Nb >::bitset(), std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset(), find(), find(), find_first_not_of(), find_first_not_of(), find_first_of(), find_last_not_of(), find_last_not_of(), find_last_of(), rfind(), and rfind().
The documentation for this class was generated from the following files:

- basic_string.h

- cow_string.h
- basic_string.tcc

6.250 std::basic_string_view< _CharT, _Traits > Class Template Reference

`#include <>>`

Public Types

- using **const_iterator**
- using **const_pointer**
- using **const_reference**
- using **const_reverse_iterator**
- using **difference_type**
- using **iterator**
- using **pointer**
- using **reference**
- using **reverse_iterator**
- using **size_type**
- using **traits_type**
- using **value_type**

Public Member Functions

- template<contiguous_iterator _It, sized_sentinel_for< _It > _End>
  constexpr **basic_string_view** (_It __first, _End __last) noexcept(noexcept(__last - __first))
- template<typename _Range, typename _DRange = remove_cvref_t<_Range>>
  constexpr **basic_string_view** (_Range &&__r) noexcept(noexcept(ranges::size(__r)) &&noexcept(ranges::↩ data(__r)))
- constexpr **basic_string_view** (const _CharT *__str) noexcept
- constexpr **basic_string_view** (const _CharT *__str, size_type __len) noexcept
- constexpr **basic_string_view** (const basic_string_view &) noexcept=default
- **basic_string_view** (nullptr_t)=delete
- constexpr const_reference **at** (size_type __pos) const
- constexpr const_reference **back** () const noexcept
- constexpr const_iterator **begin** () const noexcept
- constexpr const_iterator **cbegin** () const noexcept
- constexpr const_iterator **cend** () const noexcept
- constexpr int **compare** (basic_string_view __str) const noexcept
- constexpr int **compare** (const _CharT *__str) const noexcept
- constexpr int **compare** (size_type __pos1, size_type __n1, basic_string_view __str) const
- constexpr int **compare** (size_type __pos1, size_type __n1, basic_string_view __str, size_type __pos2, size_type __n2) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__str) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__str, size_type __n2) const noexcept(false)
- constexpr bool **contains** (_CharT __x) const noexcept
- constexpr bool **contains** (basic_string_view __x) const noexcept
- constexpr bool **contains** (const _CharT *__x) const noexcept
- constexpr size_type **copy** (_CharT *__str, size_type __n, size_type __pos=0) const
- constexpr const_reverse_iterator **crbegin** () const noexcept
- constexpr const_reverse_iterator **crend** () const noexcept

- constexpr const_pointer **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr size_type **find** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_first_of** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_last_not_of** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_not_of** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr const_reference **front** () const noexcept
- constexpr size_type **length** () const noexcept
- constexpr size_type **max_size** () const noexcept
- constexpr basic_string_view & **operator=** (const basic_string_view &) noexcept=default
- constexpr const_reference **operator[ ]** (size_type __pos) const noexcept
- constexpr const_reverse_iterator **rbegin** () const noexcept
- constexpr void **remove_prefix** (size_type __n) noexcept
- constexpr void **remove_suffix** (size_type __n) noexcept
- constexpr const_reverse_iterator **rend** () const noexcept
- constexpr size_type **rfind** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **rfind** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr size_type **size** () const noexcept
- constexpr basic_string_view **substr** (size_type __pos=0, size_type __n=npos) const noexcept(false)
- constexpr basic_string_view **subview** (size_type __pos=0, size_type __n=npos) const
- constexpr void **swap** (basic_string_view &__sv) noexcept

Static Public Attributes

- static constexpr size_type **npos**

6.250.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::basic_string_view< _CharT, _Traits >

A non-owning reference to a string.

Template Parameters

_CharT	Type of character
_Traits	Traits for character type, defaults to char_traits<_CharT>.

A basic_string_view looks like this:

```
_CharT*     _M_str
size_t      _M_len
```

The documentation for this class was generated from the following files:

- string_view
- bits/string_view.tcc

6.251 std::experimental::filesystem::v1::basic_string_view< _CharT, _Traits > Class Template Reference

`#include <>>`

Public Types

- using **const_iterator**
- using **const_pointer**
- using **const_reference**
- using **const_reverse_iterator**
- using **difference_type**
- using **iterator**
- using **pointer**
- using **reference**
- using **reverse_iterator**
- using **size_type**
- using **traits_type**
- using **value_type**

Public Member Functions

- template<contiguous_iterator _It, sized_sentinel_for< _It > _End>
  constexpr **basic_string_view** (_It __first, _End __last) noexcept(noexcept(__last - __first))
- template<typename _Range, typename _DRange = remove_cvref_t<_Range>>
  constexpr **basic_string_view** (_Range &&__r) noexcept(noexcept(ranges::size(__r)) &&noexcept(ranges::↩ data(__r)))
- constexpr **basic_string_view** (const _CharT *__str) noexcept
- constexpr **basic_string_view** (const _CharT *__str, size_type __len) noexcept
- constexpr **basic_string_view** (const basic_string_view &) noexcept=default
- **basic_string_view** (nullptr_t)=delete
- constexpr const_reference **at** (size_type __pos) const
- constexpr const_reference **back** () const noexcept
- constexpr const_iterator **begin** () const noexcept

- constexpr const_iterator **cbegin** () const noexcept
- constexpr const_iterator **cend** () const noexcept
- constexpr int **compare** (basic_string_view __str) const noexcept
- constexpr int **compare** (const _CharT *__str) const noexcept
- constexpr int **compare** (size_type __pos1, size_type __n1, basic_string_view __str) const
- constexpr int **compare** (size_type __pos1, size_type __n1, basic_string_view __str, size_type __pos2, size_type __n2) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__str) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__str, size_type __n2) const noexcept(false)
- constexpr bool **contains** (_CharT __x) const noexcept
- constexpr bool **contains** (basic_string_view __x) const noexcept
- constexpr bool **contains** (const _CharT *__x) const noexcept
- constexpr size_type **copy** (_CharT *__str, size_type __n, size_type __pos=0) const
- constexpr const_reverse_iterator **crbegin** () const noexcept
- constexpr const_reverse_iterator **crend** () const noexcept
- constexpr const_pointer **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr size_type **find** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_first_of** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_last_not_of** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_not_of** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr const_reference **front** () const noexcept
- constexpr size_type **length** () const noexcept
- constexpr size_type **max_size** () const noexcept
- constexpr basic_string_view & **operator=** (const basic_string_view &) noexcept=default
- constexpr const_reference **operator[ ]** (size_type __pos) const noexcept
- constexpr const_reverse_iterator **rbegin** () const noexcept
- constexpr void **remove_prefix** (size_type __n) noexcept
- constexpr void **remove_suffix** (size_type __n) noexcept
- constexpr const_reverse_iterator **rend** () const noexcept
- constexpr size_type **rfind** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (const _CharT *__str, size_type __pos, size_type __n) const noexcept

- constexpr size_type **rfind** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr size_type **size** () const noexcept
- constexpr basic_string_view **substr** (size_type __pos=0, size_type __n=npos) const noexcept(false)
- constexpr basic_string_view **subview** (size_type __pos=0, size_type __n=npos) const
- constexpr void **swap** (basic_string_view &__sv) noexcept

Static Public Attributes

- static constexpr size_type **npos**

6.251.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::experimental::filesystem::v1::basic_string_view< _CharT, _Traits >

A non-owning reference to a string.

Template Parameters

_CharT	Type of character
_Traits	Traits for character type, defaults to char_traits<_CharT>.

A basic_string_view looks like this:

```
_CharT*     _M_str
size_t      _M_len
```

The documentation for this class was generated from the following files:

- string_view
- bits/string_view.tcc

6.252 std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits > Class Template Reference

```
#include <>>
```

Public Types

- using **const_iterator**
- using **const_pointer**
- using **const_reference**
- using **const_reverse_iterator**
- using **difference_type**
- using **iterator**
- using **pointer**
- using **reference**
- using **reverse_iterator**
- using **size_type**
- using **traits_type**
- using **value_type**

Public Member Functions

- constexpr **basic_string_view** (const _CharT *__str)
- constexpr **basic_string_view** (const _CharT *__str, size_type __len)
- template<typename _Allocator>
  basic_string_view (const basic_string< _CharT, _Traits, _Allocator > &__str) noexcept
- constexpr **basic_string_view** (const basic_string_view &) noexcept=default
- constexpr const _CharT & **at** (size_type __pos) const
- constexpr const _CharT & **back** () const
- constexpr const_iterator **begin** () const noexcept
- constexpr const_iterator **cbegin** () const noexcept
- constexpr const_iterator **cend** () const noexcept
- constexpr int **compare** (basic_string_view __str) const noexcept
- constexpr int **compare** (const _CharT *__str) const noexcept
- constexpr int **compare** (size_type __pos1, size_type __n1, basic_string_view __str) const
- constexpr int **compare** (size_type __pos1, size_type __n1, basic_string_view __str, size_type __pos2, size_type __n2) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__str) const
- constexpr int **compare** (size_type __pos1, size_type __n1, const _CharT *__str, size_type __n2) const
- size_type **copy** (_CharT *__str, size_type __n, size_type __pos=0) const
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- constexpr const _CharT * **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr size_type **find** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **find** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_not_of** (const _CharT *__str, size_type __pos, size_type __n) const
- constexpr size_type **find_first_not_of** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (_CharT __c, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (basic_string_view __str, size_type __pos=0) const noexcept
- constexpr size_type **find_first_of** (const _CharT *__str, size_type __pos, size_type __n) const
- constexpr size_type **find_first_of** (const _CharT *__str, size_type __pos=0) const noexcept
- constexpr size_type **find_last_not_of** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_not_of** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_not_of** (const _CharT *__str, size_type __pos, size_type __n) const
- constexpr size_type **find_last_not_of** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **find_last_of** (const _CharT *__str, size_type __pos, size_type __n) const
- constexpr size_type **find_last_of** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr const _CharT & **front** () const
- constexpr size_type **length** () const noexcept
- constexpr size_type **max_size** () const noexcept
- template<typename _Allocator>
  operator basic_string< _CharT, _Traits, _Allocator > () const
- basic_string_view & **operator=** (const basic_string_view &) noexcept=default

- constexpr const _CharT & **operator[ ]** (size_type __pos) const
- const_reverse_iterator **rbegin** () const noexcept
- constexpr void **remove_prefix** (size_type __n)
- constexpr void **remove_suffix** (size_type __n)
- const_reverse_iterator **rend** () const noexcept
- constexpr size_type **rfind** (_CharT __c, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (basic_string_view __str, size_type __pos=npos) const noexcept
- constexpr size_type **rfind** (const _CharT *__str, size_type __pos, size_type __n) const noexcept
- constexpr size_type **rfind** (const _CharT *__str, size_type __pos=npos) const noexcept
- constexpr size_type **size** () const noexcept
- constexpr basic_string_view **substr** (size_type __pos=0, size_type __n=npos) const
- constexpr void **swap** (basic_string_view &__sv) noexcept
- template<typename _Allocator = std::allocator<_CharT>>
  basic_string< _CharT, _Traits, _Allocator > **to_string** (const _Allocator &__alloc=_Allocator()) const

Static Public Attributes

- static constexpr size_type **npos**

6.252.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >

A non-owning reference to a string.

Template Parameters

_CharT	Type of character
_Traits	Traits for character type, defaults to char_traits<_CharT>.

A basic_string_view looks like this:

```
_CharT*     _M_str
size_t      _M_len
```

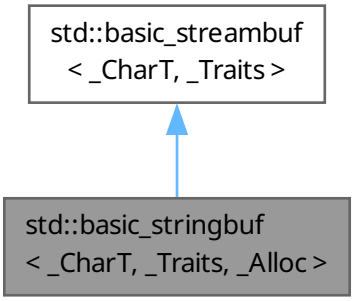
The documentation for this class was generated from the following files:

- experimental/string_view
- experimental/bits/string_view.tcc

6.253 std::basic_stringbuf< _CharT, _Traits, _Alloc > Class Template Reference

```
#include <sstream>
```

Inheritance diagram for std::basic_stringbuf< _CharT, _Traits, _Alloc >:



Public Types

- typedef __string_type::size_type **__size_type**
- typedef basic_streambuf< char_type, traits_type > **__streambuf_type**
- typedef basic_string< char_type, _Traits, _Alloc > **__string_type**
- typedef _Alloc **allocator_type**
- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**
- typedef _Traits **traits_type**

Public Member Functions

- basic_stringbuf ()
- **basic_stringbuf** (__string_type &&__s, ios_base::openmode __mode=ios_base::in|ios_base::out)
- **basic_stringbuf** (basic_stringbuf &&__rhs)
- **basic_stringbuf** (basic_stringbuf &&__rhs, const allocator_type &__a)
- basic_stringbuf (const __string_type &__str, ios_base::openmode __mode=ios_base::in|ios_base::out)
- **basic_stringbuf** (const allocator_type &__a)
- template<typename _SAlloc>
  basic_stringbuf (const basic_string< _CharT, _Traits, _SAlloc > &__s, const allocator_type &__a)
- template<typename _SAlloc>
  basic_stringbuf (const basic_string< _CharT, _Traits, _SAlloc > &__s, ios_base::openmode __mode, const allocator_type &__a)
- template<typename _SAlloc>
  basic_stringbuf (const basic_string< _CharT, _Traits, _SAlloc > &__s, ios_base::openmode __↩
  mode=ios_base::in|ios_base::out)
- **basic_stringbuf** (const basic_stringbuf &)=delete
- basic_stringbuf (ios_base::openmode __mode)
- **basic_stringbuf** (ios_base::openmode __mode, const allocator_type &__a)
- allocator_type **get_allocator** () const noexcept
- locale getloc () const

- streamsize in_avail ()
- basic_stringbuf & **operator=** (basic_stringbuf &&__rhs)
- basic_stringbuf & **operator=** (const basic_stringbuf &)=delete
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)
- __string_type **str** () &&
- __string_type str () const &
- void **str** (__string_type &&__s)
- void str (const __string_type &__s)
- template<__allocator_like _SAlloc>
  basic_string< _CharT, _Traits, _SAlloc > **str** (const _SAlloc &__sa) const
- template<__allocator_like _SAlloc>
  void **str** (const basic_string< _CharT, _Traits, _SAlloc > &__s)
- int_type sungetc ()
- void **swap** (basic_stringbuf &__rhs) noexcept(_Noexcept_swap::value)
- basic_string_view< char_type, traits_type > **view** () const noexcept

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void **_M_pbump** (char_type *__pbeg, char_type *__pend, off_type __off)
- void **_M_stringbuf_init** (ios_base::openmode __mode)
- void **_M_sync** (char_type *__base, __size_type __i, __size_type __o)
- void **_M_update_egptr** ()
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=traits_type::eof())
- virtual int_type pbackfail (int_type __c=traits_type::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual __streambuf_type * setbuf (char_type *__s, streamsize __n)
- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()

- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)

- char_type * eback () const
- char_type * gptr () const
- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- locale _M_buf_locale
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- ios_base::openmode _M_mode
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end
- __string_type **_M_string**

6.253.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_stringbuf< _CharT, _Traits, _Alloc >

The actual work of input and output (for std::string).

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

This class associates either or both of its input and output sequences with a sequence of characters, which can be initialized from, or made available as, a `std::basic_string`. (Paraphrased from [27.7.1]/1.)
For this class, open modes (of type `ios_base::openmode`) have `in` set if the input sequence can be read, and `out` set if the output sequence can be written.

6.253.2 Constructor & Destructor Documentation

basic_stringbuf() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf ()  [inline]
```

Starts with an empty string buffer.
The default constructor initializes the parent class using its own default ctor.

basic_stringbuf() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (
            ios_base::openmode __mode)  [inline], [explicit]
```

Starts with an empty string buffer.

Parameters

__mode	Whether the buffer can read, or write, or both.

The default constructor initializes the parent class using its own default ctor.

basic_stringbuf() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringbuf< _CharT, _Traits, _Alloc >::basic_stringbuf (
            const __string_type & __str,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [explicit]
```

Starts with an existing string buffer.

Parameters

__str	A string to copy as a starting buffer.
__mode	Whether the buffer can read, or write, or both.

This constructor initializes the parent class using its own default ctor.

6.253.3 Member Function Documentation

eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _Cha std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_streambuf< char_type, traits_type >::in_avail std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< char std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sputc(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_streambuf< char_type, traits_type >::uflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const  [inline], [inherited]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.
Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy
std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_stringbuf< _CharT,
std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< cha
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_streambuf< char_type, traits_type >::sgetc(),
std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< char >::showmanyc(), std::basic_streambuf< char_type, traits
std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_streambuf< char_type, traits_type >::uflow(),
std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(),
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and
xsgetn().

imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
             const locale & __loc)  [inline], [protected], [virtual], [inherited]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >,
std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >,
and std::basic_filebuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

in_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ()  [inline], [inherited]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

overflow()

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↩
Alloc >::overflow (
            int_type __c = traits_type::eof())  [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

_↩ _c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually __c, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if __c is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::basic_string< _CharT, _Traits, _Alloc >::assign(), std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::epptr(), std::basic_streambuf< _CharT, _Tra std::ios_base::in, std::max(), std::min(), std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase(), std::basic_streambuf< _CharT, _Traits >::pbump(), std::basic_streambuf< _CharT, _Traits >::pptr(), std::basic_string< _CharT, _Traits, std::basic_string< _CharT, _Traits, _Alloc >::reserve(), and std::basic_streambuf< _CharT, _Traits >::setg().

pbackfail()

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↩
Alloc >::pbackfail (
            int_type __c = traits_type::eof())  [protected], [virtual]
```

Tries to back up the input sequence.

Parameters

_↩ _c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::gbump(), std::basic_streambuf< _CharT, _Traits >::gptr(), and std::ios_base::out.

pbase()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, std::basic_stringbuf< char >::str(), std::basic_filebuf< _CharT, _Traits >::sync(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), and std::basic_streambuf< char_type, traits_type >::sputc().

pptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_streambuf< char_t std::basic_filebuf< _CharT, _Traits >::sync(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
            const locale & __loc)  [inline], [inherited]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
```

```
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ()  [inline], [inherited]
```

Calls virtual sync function.
Referenced by std::basic_istream< _CharT, _Traits >::sync().

sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_streambuf< char_type, traits_type >::snextc().

seekoff()

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type std::basic_stringbuf< _CharT, _Traits, _↩
Alloc >::seekoff (
            off_type ,
            ios_base::seekdir ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [protected], [virtual]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::ios_base::cur, std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::e
std::ios_base::end, std::basic_streambuf< _CharT, _Traits >::epptr(), std::basic_streambuf< _CharT, _Traits >::gptr(),
std::ios_base::in, std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase(), std::basic_streambuf< _CharT, _Traits >::pptr(),
and std::basic_streambuf< _CharT, _Traits >::setg().

seekpos()

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::pos_type std::basic_stringbuf< _CharT, _Traits, _↩
Alloc >::seekpos (
            pos_type ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [protected], [virtual]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::egptr(),
std::basic_streambuf< _CharT, _Traits >::epptr(), std::ios_base::in, std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase()
and std::basic_streambuf< _CharT, _Traits >::setg().

setbuf()

```
template<typename _CharT, typename _Traits, typename _Alloc>
virtual __streambuf_type * std::basic_stringbuf< _CharT, _Traits, _Alloc >::setbuf (
            char_type * __s,
            streamsize __n)  [inline], [protected], [virtual]
```

Manipulates the buffer.

Parameters

_↩ _s	Pointer to a buffer area.
_↩ _n	Size of __s.

Returns

`this`

If no buffer has already been created, and both __s and __n are non-zero, then __s is used as a buffer; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.↩ buffering for more.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.

setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
```

```
            char_type * __gbeg,
            char_type * __gnext,
            char_type * __gend)  [inline], [protected], [inherited]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), std::basic_filebuf< char_type, traits_type >::_M_create
std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_filebuf< char_type, traits_type >::_M_set_buffer(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
            char_type * __pbeg,
            char_type * __pend)  [inline], [protected], [inherited]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), and std::basic_filebuf< char_type, traits_type >::_M_s

sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::get(),
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ign
and std::basic_streambuf< char_type, traits_type >::snextc().

sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for xsgetn.

Parameters

_↩ _s	A buffer area.
_↩ _n	A count.

Returns xsgetn(__s,__n). The effect is to fill __s[0] through __s[__n-1] with characters from the input sequence, if possible.

showmanyc()

```
template<typename _CharT, typename _Traits, typename _Alloc>
virtual streamsize std::basic_stringbuf< _CharT, _Traits, _Alloc >::showmanyc ()  [inline], [protected],
[virtual]
```

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, char_traits< _CharT > >::get(), and std::basic_istream< _CharT, char_traits< _CharT > >::readsome().

sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
            char_type __c)  [inline], [inherited]
```

Pushing characters back into the input stream.

Parameters

_↩ _c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but __c is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be __c.
Referenced by std::basic_istream< _CharT, _Traits >::putback().

sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
            char_type __c)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _c	A character to output.

Returns

__c, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.
Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow().

sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
            const char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _s	A buffer read area.
_↩ _n	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __s[0] through __s[__n-1] to the output sequence, if possible.

str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_stringbuf< _CharT, _Traits, _Alloc >::str () const &  [inline], [nodiscard]
```

Copying out the string buffer.

Returns

A copy of one of the underlying sequences.

If the buffer is only created in input mode, the underlying character sequence is equal to the input sequence; otherwise, it is equal to the output sequence. [27.7.1.2]/1

str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_stringbuf< _CharT, _Traits, _Alloc >::str (
             const __string_type & __s)  [inline]
```

Setting a new buffer.

Parameters

_↩ _s	The string to use as a new sequence.

Deallocates any previous stored sequence, then copies *s* to use as a new one.

sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ()  [inline], [inherited]
```

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.
Referenced by std::basic_istream< _CharT, _Traits >::unget().

sync()

```
template<typename _CharT, typename _Traits>
virtual int std::basic_streambuf< _CharT, _Traits >::sync ()  [inline], [protected], [virtual],
[inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, and std::wbuffer_convert< _Codecvt, _Elem, _Tr >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubsync().

uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ()  [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.
Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.
Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc(), and xsgetn().

underflow()

```
template<class _CharT, class _Traits, class _Alloc>
basic_stringbuf< _CharT, _Traits, _Alloc >::int_type std::basic_stringbuf< _CharT, _Traits, _↩
Alloc >::underflow ()  [protected], [virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::gptr(), and std::ios_base::in.

xsgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsgetn (
            char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character extraction.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __s[0] through __s[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __n characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, and std::basic_filebuf< char_type, traits_type >.
References egptr(), gptr(), std::min(), and uflow().
Referenced by std::basic_streambuf< char_type, traits_type >::sgetn().

xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character insertion.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __s[0] through __s[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, and std::basic_filebuf< char_type, traits_type >.
References epptr(), std::min(), overflow(), and pptr().
Referenced by std::basic_streambuf< char_type, traits_type >::sputn().

6.253.4 Member Data Documentation

_M_buf_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale  [protected], [inherited]
```

Current locale setting.
Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

_M_in_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg  [protected], [inherited]
```

Start of get area.

_M_in_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur  [protected], [inherited]
```

Current read area.

_M_in_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end  [protected], [inherited]
```

End of get area.

_M_mode

```
template<typename _CharT, typename _Traits, typename _Alloc>
ios_base::openmode std::basic_stringbuf< _CharT, _Traits, _Alloc >::_M_mode  [protected]
```

Place to stash in || out || in | out settings for current stringbuf.
Referenced by overflow(), pbackfail(), seekoff(), seekpos(), and underflow().

_M_out_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg  [protected], [inherited]
```

Start of put area.

_M_out_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur  [protected], [inherited]
```

Current put area.

_M_out_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end  [protected], [inherited]
```

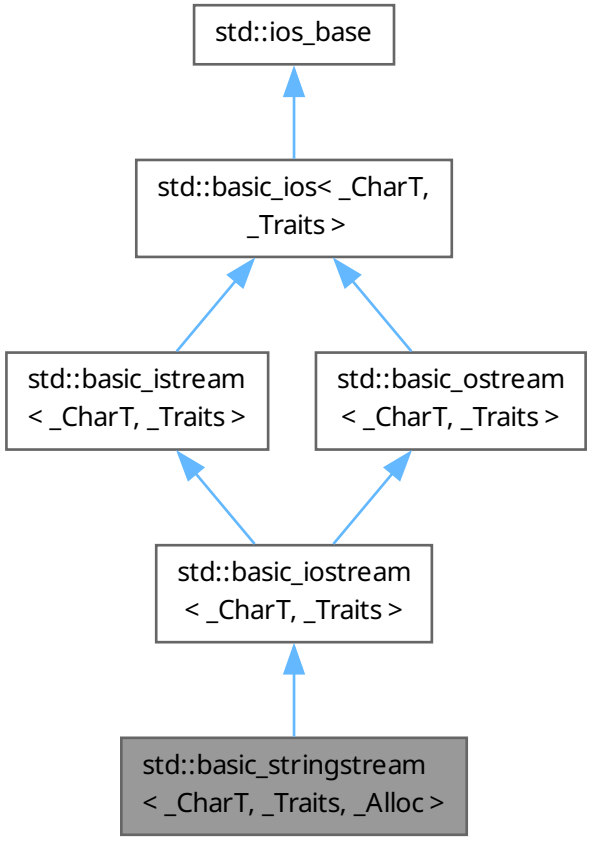
End of put area.
The documentation for this class was generated from the following files:

- sstream
- sstream.tcc

6.254 std::basic_stringstream< _CharT, _Traits, _Alloc > Class Template Reference

`#include <sstream>`

Inheritance diagram for std::basic_stringstream< _CharT, _Traits, _Alloc >:



Public Types

- typedef ctype< _CharT > **__ctype_type**
- typedef ctype< _CharT > **__ctype_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_ios< _CharT, _Traits > **__ios_type**
- typedef basic_iostream< char_type, traits_type > **__iostream_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef num_get< _CharT, istreambuf_iterator< _CharT, _Traits > > **__num_get_type**
- typedef num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > > **__num_put_type**
- typedef basic_ostream< _CharT, _Traits > **__ostream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef basic_string< _CharT, _Traits, _Alloc > **__string_type**

- typedef basic_stringbuf< _CharT, _Traits, _Alloc > **__stringbuf_type**
- typedef _Alloc **allocator_type**
- typedef _CharT **char_type**
- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(* event_callback) (event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags
- typedef traits_type::int_type **int_type**
- typedef _Ios_Iostate iostate
- typedef traits_type::off_type **off_type**
- typedef _Ios_Openmode openmode
- typedef traits_type::pos_type **pos_type**
- typedef _Ios_Seekdir seekdir
- typedef _Traits **traits_type**

Public Member Functions

- basic_stringstream ()
- **basic_stringstream** (__string_type &&__str, ios_base::openmode __mode=ios_base::in|ios_base::out)
- **basic_stringstream** (basic_stringstream &&__rhs)
- basic_stringstream (const __string_type &__str, ios_base::openmode __m=ios_base::out|ios_base::in)
- template<typename _SAlloc>
  basic_stringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, const allocator_type &__a)
- template<typename _SAlloc>
  basic_stringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, ios_base::openmode __mode, const allocator_type &__a)
- template<typename _SAlloc>
  basic_stringstream (const basic_string< _CharT, _Traits, _SAlloc > &__str, ios_base::openmode __↩ mode=ios_base::in|ios_base::out)
- **basic_stringstream** (const basic_stringstream &)=delete
- basic_stringstream (ios_base::openmode __m)
- **basic_stringstream** (ios_base::openmode __mode, const allocator_type &__a)
- ~basic_stringstream ()
- template<typename _ValueT>
  basic_istream< _CharT, _Traits > & **_M_extract** (_ValueT &__v)
- const locale & _M_getloc () const
- template<typename _ValueT>
  basic_ostream< _CharT, _Traits > & **_M_insert** (_ValueT __v)
- void **_M_setstate** (iostate __state)
- bool bad () const
- void clear (iostate __state=goodbit)
- basic_ios & copyfmt (const basic_ios &__rhs)
- bool eof () const
- iostate exceptions () const
- void exceptions (iostate __except)
- bool fail () const
- char_type fill () const
- char_type fill (char_type __ch)
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- __ostream_type & flush ()
- streamsize gcount () const

- basic_istream< wchar_t > & **getline** (char_type *__s, streamsize __n, char_type __delim)
- basic_istream< char > & getline (char_type *__s, streamsize __n, char_type __delim)
- locale getloc () const
- bool good () const
- basic_istream< wchar_t > & **ignore** (streamsize __n)
- basic_istream< char > & **ignore** (streamsize __n)
- basic_istream< wchar_t > & **ignore** (streamsize __n, int_type __delim)
- basic_istream< char > & **ignore** (streamsize __n, int_type __delim)
- locale imbue (const locale &__loc)
- long & iword (int __ix)
- char narrow (char_type __c, char __dfault) const
- __ostream_type & operator<< (__streambuf_type *__sb)
- __ostream_type & operator<< (const void *__p)
- __ostream_type & **operator**<< (const volatile void *__p)
- __ostream_type & **operator**<< (nullptr_t)
- basic_stringstream & **operator=** (basic_stringstream &&__rhs)
- basic_stringstream & **operator=** (const basic_stringstream &)=delete
- __istream_type & operator>> (__streambuf_type *__sb)
- __istream_type & operator>> (void *&__p)
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- basic_streambuf< _CharT, _Traits > * rdbuf (basic_streambuf< _CharT, _Traits > *__sb)
- __stringbuf_type * rdbuf () const
- iostate rdstate () const
- void register_callback (event_callback __fn, int __index)
- __ostream_type & seekp (off_type, ios_base::seekdir)
- __ostream_type & seekp (pos_type)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void setstate (iostate __state)
- __string_type **str** () &&
- __string_type str () const &
- void **str** (__string_type &&__s)
- void str (const __string_type &__s)
- template<__allocator_like _SAlloc>
  basic_string< _CharT, _Traits, _SAlloc > **str** (const _SAlloc &__sa) const
- template<__allocator_like _SAlloc>
  void **str** (const basic_string< _CharT, _Traits, _SAlloc > &__s)
- void **swap** (basic_stringstream &__rhs)
- pos_type tellp ()
- basic_ostream< _CharT, _Traits > * tie () const
- basic_ostream< _CharT, _Traits > * tie (basic_ostream< _CharT, _Traits > *__tiestr)
- void unsetf (fmtflags __mask)
- basic_string_view< char_type, traits_type > **view** () const noexcept
- char_type widen (char __c) const
- streamsize width () const
- streamsize width (streamsize __wide)

- __istream_type & operator>> (__istream_type &(*__pf)(__istream_type &))

- __istream_type & operator>> (__ios_type &(*__pf)(__ios_type &))
- __istream_type & operator>> (ios_base &(*__pf)(ios_base &))

Extractors

All the `operator>>` functions (aka formatted input functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to false. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- __istream_type & operator>> (bool &__n)
- __istream_type & operator>> (short &__n)
- __istream_type & operator>> (unsigned short &__n)
- __istream_type & operator>> (int &__n)
- __istream_type & operator>> (unsigned int &__n)
- __istream_type & operator>> (long &__n)
- __istream_type & operator>> (unsigned long &__n)
- __istream_type & operator>> (long long &__n)
- __istream_type & operator>> (unsigned long long &__n)

- __istream_type & operator>> (float &__f)
- __istream_type & operator>> (double &__f)
- __istream_type & operator>> (long double &__f)

Unformatted Input Functions

All the unformatted input functions have some common behavior. Each starts by constructing a temporary object of type std::basic_istream::sentry with the second argument (noskipws) set to true. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to extract whatever data is appropriate for the type of the argument.

The number of characters extracted is stored for later retrieval by gcount().

If an exception is thrown during extraction, ios_base::badbit will be turned on in the stream's error state (without causing an ios_base::failure to be thrown) and the original exception will be rethrown if badbit is set in the exceptions mask.

- int_type get ()
- __istream_type & get (char_type &__c)
- __istream_type & get (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & get (char_type *__s, streamsize __n)
- __istream_type & get (__streambuf_type &__sb, char_type __delim)
- __istream_type & get (__streambuf_type &__sb)
- __istream_type & getline (char_type *__s, streamsize __n, char_type __delim)
- __istream_type & getline (char_type *__s, streamsize __n)
- __istream_type & ignore (streamsize __n, int_type __delim)
- __istream_type & ignore (streamsize __n)
- __istream_type & ignore ()
- __istream_type & ignore (streamsize __n, char __delim)
- int_type peek ()
- __istream_type & read (char_type *__s, streamsize __n)
- streamsize readsome (char_type *__s, streamsize __n)
- __istream_type & putback (char_type __c)
- __istream_type & unget ()
- int sync ()
- pos_type tellg ()

- __istream_type & seekg (pos_type)
- __istream_type & seekg (off_type, ios_base::seekdir)

- operator bool () const
- bool operator! () const

- __ostream_type & operator<< (__ostream_type &(*__pf)(__ostream_type &))
- __ostream_type & operator<< (__ios_type &(*__pf)(__ios_type &))
- __ostream_type & operator<< (ios_base &(*__pf)(ios_base &))

Inserters

All the `operator<<` functions (aka formatted output functions) have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This can have several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state without causing an ios_base::failure to be thrown. The original exception will then be rethrown.

- __ostream_type & operator<< (long __n)
- __ostream_type & operator<< (unsigned long __n)
- __ostream_type & operator<< (bool __n)
- __ostream_type & operator<< (short __n)
- __ostream_type & operator<< (unsigned short __n)
- __ostream_type & operator<< (int __n)
- __ostream_type & operator<< (unsigned int __n)
- __ostream_type & operator<< (long long __n)
- __ostream_type & operator<< (unsigned long long __n)

- __ostream_type & operator<< (double __f)
- __ostream_type & operator<< (float __f)
- __ostream_type & operator<< (long double __f)

Unformatted Output Functions

All the unformatted output functions have some common behavior. Each starts by constructing a temporary object of type std::basic_ostream::sentry. This has several effects, concluding with the setting of a status flag; see the sentry documentation for more.

If the sentry status is good, the function tries to generate whatever data is appropriate for the type of the argument.

If an exception is thrown during insertion, ios_base::badbit will be turned on in the stream's error state. If badbit is on in the stream's exceptions mask, the exception will be rethrown without completing its actions.

- __ostream_type & put (char_type __c)
- __ostream_type & write (const char_type *__s, streamsize __n)

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal
- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_cache_locale** (const locale &__loc)
- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- template<typename _ValueT>
  __istream_type & **_M_extract** (_ValueT &__v)
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- template<typename _ValueT>
  __ostream_type & **_M_insert** (_ValueT __v)
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept

- void init (basic_streambuf< _CharT, _Traits > *__sb)
- void **move** (basic_ios &&__rhs)
- void **move** (basic_ios &__rhs)
- void **set_rdbuf** (basic_streambuf< _CharT, _Traits > *__sb)
- void **swap** (basic_ios &__rhs) noexcept
- void **swap** (basic_iostream &__rhs)
- void **swap** (basic_istream &__rhs)
- void **swap** (basic_ostream &__rhs)

Protected Attributes

- _Callback_list * **_M_callbacks**
- const __ctype_type * **_M_ctype**
- iostate **_M_exception**
- char_type **_M_fill**
- bool **_M_fill_init**
- fmtflags **_M_flags**
- streamsize _M_gcount
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- const __num_get_type * **_M_num_get**
- const __num_put_type * **_M_num_put**
- streamsize **_M_precision**
- basic_streambuf< _CharT, _Traits > * **_M_streambuf**
- iostate **_M_streambuf_state**
- basic_ostream< _CharT, _Traits > * **_M_tie**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

6.254.1 Detailed Description

template<typename _CharT, typename _Traits, typename _Alloc>
class std::basic_stringstream< _CharT, _Traits, _Alloc >

Controlling input and output for std::string.

Template Parameters

_CharT	Type of character stream.
_Traits	Traits for character type, defaults to char_traits<_CharT>.
_Alloc	Allocator type, defaults to allocator<_CharT>.

This class supports reading from and writing to objects of type std::basic_string, using the inherited functions from std::basic_iostream. To control the associated sequence, an instance of std::basic_stringbuf is used, which this page refers to as `sb`.

6.254.2 Member Typedef Documentation

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)  [inherited]
```

The type of an event callback function.

Parameters

_↩ _e	One of the members of the event enum.
_↩ _b	Reference to the ios_base object.
_↩ _i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ∼ios().

fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags  [inherited]
```

This is a bitmask type.
_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield

- basefield
- floatfield

iostate

`typedef _Ios_Iostate std::ios_base::iostate [inherited]`
This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

`typedef _Ios_Openmode std::ios_base::openmode [inherited]`
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

`typedef _Ios_Seekdir std::ios_base::seekdir [inherited]`
This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

6.254.3 Member Enumeration Documentation

event

`enum std::ios_base::event [inherited]`
The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

6.254.4 Constructor & Destructor Documentation

basic_stringstream() [1/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream ()  [inline]
```

Default constructor starts with an empty string buffer.
Initializes sb using the mode in|out, and passes &sb to the base class initializer. Does not allocate any buffer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

basic_stringstream() [2/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream (
            ios_base::openmode __m)  [inline], [explicit]
```

Starts with an empty string buffer.

Parameters

_↩ _m	Whether the buffer can read, or write, or both.

Initializes sb using the mode from __m, and passes &sb to the base class initializer. Does not allocate any buffer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

basic_stringstream() [3/3]

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream< _CharT, _Traits, _Alloc >::basic_stringstream (
            const __string_type & __str,
            ios_base::openmode __m = ios_base::out | ios_base::in)  [inline], [explicit]
```

Starts with an existing string buffer.

Parameters

__str	A string to copy as a starting buffer.
__m	Whether the buffer can read, or write, or both.

Initializes sb using __str and __m, and passes &sb to the base class initializer.
That's a lie. We initialize the base class with NULL, because the string class does its own memory management.

~basic_stringstream()

```
template<typename _CharT, typename _Traits, typename _Alloc>
std::basic_stringstream< _CharT, _Traits, _Alloc >::~basic_stringstream ()  [inline]
```

The destructor does nothing.
The buffer is deallocated by the stringbuf object, not the formatting stream.

6.254.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline], [inherited]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

bad()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::bad () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the badbit is set.

Note that other iostate flags may also be set.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry().

clear()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::clear (
             iostate __state = goodbit)  [inherited]
```

[Re]sets the error state.

Parameters

__state	The new state flag(s) to set.

See std::ios_base::iostate for the possible bit values. Most users will not need to pass an argument.
References std::ios_base::badbit, exceptions(), rdbuf(), and rdstate().
Referenced by std::basic_ios< char_type, traits_type >::exceptions(), std::basic_fstream< char >::open(), std::basic_fstream< char >::
std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::__detail::operator>>(), std::basic_istream< _CharT, _Traits >::putback(),
rdbuf(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits
and std::basic_istream< _CharT, _Traits >::unget().

copyfmt()

```
template<typename _CharT, typename _Traits>
basic_ios< _CharT, _Traits > & std::basic_ios< _CharT, _Traits >::copyfmt (
             const basic_ios< _CharT, _Traits > & __rhs)  [inherited]
```

Copies fields of __rhs into this.

Parameters

__rhs	The source values for the copies.

Returns

Reference to this object.

All fields of __rhs are copied into this object except that rdbuf() and rdstate() remain unchanged. All values in the pword and iword arrays are copied. Before copying, each callback is invoked with erase_event. After copying, each (new) callback is invoked with copyfmt_event. The final step is to copy exceptions().
References basic_ios(), std::__addressof(), exceptions(), fill(), std::ios_base::flags(), std::ios_base::getloc(), std::ios_base::precision(), tie(), std::tie(), and std::ios_base::width().

eof()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::eof () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if the eofbit is set.

Note that other iostate flags may also be set.

exceptions() [1/2]

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::exceptions () const  [inline], [nodiscard], [inherited]
```

Throwing exceptions on errors.

Returns

The current exceptions mask.

This changes nothing in the stream. See the one-argument version of exceptions(iostate) for the meaning of the return value.
Referenced by clear(), and copyfmt().

exceptions() [2/2]

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::exceptions (
            iostate __except)  [inline], [inherited]
```

Throwing exceptions on errors.

Parameters

__except	The new exceptions mask.

By default, error flags are set silently. You can set an exceptions mask for each stream; if a bit in the mask becomes set in the error flags, then an exception of type std::ios_base::failure is thrown.
If the error flag is already set when the exceptions mask is added, the exception is immediately thrown. Try running the following under GCC 3.1 or later:

```
#include <iostream>
#include <fstream>
#include <exception>

int main()
{
    std::set_terminate (__gnu_cxx::__verbose_terminate_handler);

    std::ifstream f ("/etc/motd");

    std::cerr « "Setting badbit\n";
```

```
    f.setstate (std::ios_base::badbit);

    std::cerr « "Setting exception mask\n";
    f.exceptions (std::ios_base::badbit);
}
```

fail()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::fail () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if either the badbit or the failbit is set.

Checking the badbit in fail() is historical practice. Note that other iostate flags may also be set.
Referenced by std::basic_ios< char_type, traits_type >::operator bool(), std::basic_ios< char_type, traits_type >::operator!(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits

fill() [1/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill () const  [inline], [nodiscard], [inherited]
```

Retrieves the *empty* character.

Returns

The current fill character.

It defaults to a space (' ') in the current locale.
Referenced by copyfmt(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::o

fill() [2/2]

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::fill (
             char_type __ch)  [inline], [inherited]
```

Sets a new *empty* character.

Parameters

__ch	The new character.

Returns

The previous fill character.

The fill character is used to fill out space when P+ characters have been requested (e.g., via setw), Q characters are actually used, and Q<P. It defaults to a space (' ') in the current locale.

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard], [inherited]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<,
std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

flush()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::flush ()  [inherited]
```

Synchronizing the stream buffer.

Returns

*this

If `rdbuf()` is a null pointer, changes nothing.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit.
References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits
Referenced by std::flush().

gcount()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::gcount () const  [inline], [inherited]
```

Character counting.

Returns

The number of characters extracted by the previous unformatted input function dispatched for this stream.

get() [1/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::get (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().
Referenced by std::basic_istream< char_type, traits_type >::get(), and std::basic_istream< char_type, traits_type >::get().

get() [2/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb)  [inline], [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.

Returns

*this

Returns `get`(__sb,widen('\n')).

get() [3/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            __streambuf_type & __sb,
            char_type __delim)  [inherited]
```

Extraction into another streambuf.

Parameters

__sb	A streambuf in which to store data.
__delim	A "stop" character.

Returns

*this

Characters are extracted and inserted into *__sb* until one of the following happens:

- the input sequence reaches EOF
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted)
- the next character equals *__delim* (in this case, the character is not extracted)
- an exception occurs (and in this case is caught)

If no characters are stored, failbit is set in the stream's error state.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

get() [4/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type & __c)  [inherited]
```

Simple extraction.

Parameters

_↩ _c	The character in which to store data.

Returns

*this

Tries to extract a character and store it in __c. If none are available, sets failbit and returns traits::eof().

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

get() [5/6]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Simple multiple-character extraction.

Parameters

_↩ _s	Pointer to an array.
_↩ _n	Maximum number of characters to store in *s*.

Returns

*this

Returns get(__s,__n,widen('\n')).

get() [6/6]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::get (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Simple multiple-character extraction.

Parameters

__s	Pointer to an array.
__n	Maximum number of characters to store in *__s*.
__delim	A "stop" character.

Returns

*this

Characters are extracted and stored into *__s* until one of the following happens:

- `__n-1` characters are stored
- the input sequence reaches EOF
- the next character equals *__delim*, in which case the character is not extracted

If no characters are stored, failbit is set in the stream's error state.
In any case, a null character is stored into the next location in the array.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_streambuf< _CharT, _Traits >::sgetc(), and std::basic_streambuf< _CharT, _Traits >::snextc().

getline() [1/3]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

String extraction.

Parameters

_↩ _s	A character array in which to store the data.
_↩ _n	Maximum number of characters to extract.

Returns

*this

Returns `getline(__s,__n,widen('\n'))`.

getline() [2/3]

```
basic_istream< char > & std::basic_istream< char >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

Explicit specialization declarations, defined in src/istream.cc.

getline() [3/3]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::getline (
            char_type * __s,
            streamsize __n,
            char_type __delim)  [inherited]
```

String extraction.

Parameters

__s	A character array in which to store the data.
__n	Maximum number of characters to extract.
__delim	A "stop" character.

Returns

*this

Extracts and stores characters into *__s* until one of the following happens. Note that these criteria are required to be tested in the order listed here, to allow an input line to exactly fill the *__s* array without setting failbit.

1. the input sequence reaches end-of-file, in which case eofbit is set in the stream error state
2. the next character equals `__delim`, in which case the character is extracted (and therefore counted in `gcount()`) but not stored
3. `__n-1` characters are stored, in which case failbit is set in the stream error state

If no characters are extracted, failbit is set. (An empty line of input should therefore not cause failbit to be set.)
In any case, a null character is stored in the next location in the array.
References _M_gcount, and std::ios_base::goodbit.
Referenced by std::basic_istream< char_type, traits_type >::getline().

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard], [inherited]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

good()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::good () const  [inline], [nodiscard], [inherited]
```

Fast error checking.

Returns

True if no error flags are set.

A wrapper around rdstate.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and std::__detail::operator>>().

ignore() [1/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            void )  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >::sbumpc().
Referenced by std::basic_istream< char_type, traits_type >::ignore().

ignore() [2/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n)  [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

ignore() [3/4]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            char __delim)  [inline], [inherited]
```

Simple extraction.

Returns

A character, or eof().

Tries to extract a character. If none are available, sets failbit and returns traits::eof().

ignore() [4/4]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::ignore (
            streamsize __n,
            int_type __delim)  [inherited]
```

Discarding characters.

Parameters

__n	Number of characters to discard.
__delim	A "stop" character.

Returns

*this

Extracts characters and throws them away until one of the following happens:

- if *__n* != `std::numeric_limits<int>::max()`, *__n* characters are extracted
- the input sequence reaches end-of-file
- the next character equals *__delim* (in this case, the character is extracted); note that this condition will never occur if *__delim* equals `traits::eof()`.

NB: Provide four overloads, instead of the single function (with defaults) mandated by the Standard: this leads to a better performing implementation, while still conforming to the Standard.
References _M_gcount, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_streambuf< _CharT, _Traits >:

imbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_ios< _CharT, _Traits >::imbue (
            const locale & __loc)  [inherited]
```

Moves to a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls `ios_base::imbue(loc)`, and if a stream buffer is associated with this stream, calls that buffer's `pubimbue(loc)`.
Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩
html
References std::ios_base::getloc(), std::ios_base::imbue(), and rdbuf().
Referenced by std::chrono::operator<<().

init()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::init (
            basic_streambuf< _CharT, _Traits > * __sb)  [protected], [inherited]
```

All setup is performed here.
This is called from the public constructor. It is not virtual and cannot be redefined.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_ios< char_type, traits_type >::basic_ios(), std::basic_istream< char_type, traits_type >::basic_istream(),

std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(), std::basic_istringstream< char >::basic_istringstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ostream< char_type, traits_type >::basic_ostream(), std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(), std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringstream< char >::basic_stringstream(), std::basic_stringstream< char >::basic_stringstream(), and std::basic_stringstream< char >::basic_stringstream().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline], [inherited]
```

Access to integer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

narrow()

```
template<typename _CharT, typename _Traits>
char std::basic_ios< _CharT, _Traits >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Squeezes characters.

Parameters

__c	The character to narrow.
__dfault	The character to narrow.

Returns

The narrowed character.

Maps a character of `char_type` to a character of `char`, if possible.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).narrow(c,dfault)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.↩ html

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ios< _CharT, _Traits >::operator bool () const  [inline], [explicit], [nodiscard],
[inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator"!()

```
template<typename _CharT, typename _Traits>
bool std::basic_ios< _CharT, _Traits >::operator!  () const  [inline], [nodiscard], [inherited]
```

The quick-and-easy status check.
This allows you to write constructs such as `if (!a_stream) ...` and `while (a_stream) ...`

operator<<() [1/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [2/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            __ostream_type &(* __pf )(__ostream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as `std::endl` and `std::hex` use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [3/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            __streambuf_type * __sb)  [inherited]
```

Extracting from another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If `__sb` is NULL, the stream will set failbit in its error state.
Characters are extracted from `__sb` and inserted into `*this` until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output sequence fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs while getting a character from `__sb`, which sets failbit in the error state

If the function inserts no characters, failbit is set.

operator<<() [4/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            bool __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [5/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            const void * __p)  [inline], [inherited]
```

Pointer arithmetic inserters.

Parameters

_↩ _p	A variable of pointer type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [6/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [7/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            float __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [8/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            int __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [9/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::endl and std::hex use these functions in constructs like "std::cout << std::endl". For more information, see the iomanip header.

operator<<() [10/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [11/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long double __f)  [inline], [inherited]
```

Floating point arithmetic inserters.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [12/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [13/17]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::operator<< (
            short __n)  [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.
References basic_ostream(), std::ios_base::badbit, std::ostreambuf_iterator< _CharT, _Traits >::failed(), std::ios_base::goodbit, std::num_put< _CharT, _OutIter >::put(), std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator<<() [14/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned int __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [15/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [16/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned long long __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator<<() [17/17]

```
template<typename _CharT, typename _Traits>
__ostream_type & std::basic_ostream< _CharT, _Traits >::operator<< (
            unsigned short __n)  [inline], [inherited]
```

Integer arithmetic inserters.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to perform numeric formatting.

operator>>() [1/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __ios_type &(* __pf )(__ios_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [2/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            __istream_type &(* __pf )(__istream_type &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [3/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            __streambuf_type * __sb)  [inherited]
```

Extracting into another streambuf.

Parameters

__sb	A pointer to a streambuf

This function behaves like one of the basic arithmetic extractors, in that it also constructs a sentry object and has the same error handling behavior.
If __sb is NULL, the stream will set failbit in its error state.
Characters are extracted from this stream and inserted into the __sb streambuf until one of the following occurs:

- the input stream reaches end-of-file,
- insertion into the output buffer fails (in this case, the character that would have been inserted is not extracted), or
- an exception occurs (and in this case is caught)

If the function inserts no characters, failbit is set.
References std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

operator>>() [4/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            bool & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [5/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [6/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            float & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ *f*	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [7/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            int & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ *_n*	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [8/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            ios_base &(* __pf )(ios_base &))  [inline], [inherited]
```

Interface for manipulators.
Manipulators such as std::ws and std::dec use these functions in constructs like std::cin >> std::ws. For more information, see the iomanip header.

operator>>() [9/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [10/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long double & __f)  [inline], [inherited]
```

Floating point arithmetic extractors.

Parameters

↩ _↩ ↩ _↩ f	A variable of builtin floating point type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [11/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [12/17]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::operator>> (
            short & __n)  [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.
References std::ios_base::badbit, std::ios_base::failbit, std::num_get< _CharT, _InIter >::get(), std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::setstate(), and std::use_facet().

operator>>() [13/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned int & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [14/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [15/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned long long & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [16/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            unsigned short & __n)  [inline], [inherited]
```

Integer arithmetic extractors.

Parameters

_↩ _n	A variable of builtin integral type.

Returns

*this if successful

These functions use the stream's current locale (specifically, the num_get facet) to parse the input data.

operator>>() [17/17]

```
template<typename _CharT, typename _Traits>
__istream_type & std::basic_istream< _CharT, _Traits >::operator>> (
            void *& __p)  [inline], [inherited]
```

Basic arithmetic extractors.

Parameters

_↩ _p	A variable of pointer type.

Returns

`*this` if successful

These functions use the stream's current locale (specifically, the `num_get` facet) to parse the input data.

peek()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::int_type std::basic_istream< _CharT, _Traits >::peek (
            void )  [inherited]
```

Looking ahead in the stream.

Returns

The next character, or eof().

If, after constructing the sentry object, `good()` is false, returns `traits::eof()`. Otherwise reads but does not extract the next input character.

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline], [inherited]
```

Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

put()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::put (
            char_type __c)  [inherited]
```

Simple insertion.

Parameters

_↩ _c	The character to insert.

Returns

*this

Tries to insert __c.

Note

This function is not overloaded on signed char and unsigned char.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits
Referenced by std::endl(), and std::ends().

putback()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::putback (
             char_type __c)  [inherited]
```

Unextracting a single character.

Parameters

_↩ _c	The character to push back into the input stream.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sputbackc(c)`.
If `rdbuf()` is null or if `sputbackc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sputbackc().
Referenced by std::operator<<().

pword()

```
void *& std::ios_base::pword (
             int __ix)  [inline], [inherited]
```

Access to void pointer array.

Parameters

_↩ _*ix*	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

rdbuf() [1/2]

```
template<typename _CharT, typename _Traits>
basic_streambuf< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::rdbuf (
            basic_streambuf< _CharT, _Traits > * __sb)  [inherited]
```

Changing the underlying buffer.

Parameters

__sb	The new stream buffer.

Returns

The previous stream buffer.

Associates a new buffer with the current stream, and clears the error state.
Due to historical accidents which the LWG refuses to correct, the I/O library suffers from a design error: this function is hidden in derived classes by overrides of the zero-argument `rdbuf()`, which is non-virtual for hysterical raisins. As a result, you must use explicit qualifications to access this function via any derived class. For example:

```
std::fstream     foo;         // or some other derived type
std::streambuf*  p = .....;

foo.ios::rdbuf(p);            // ios == basic_ios<char>
```

References clear().

rdbuf() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__stringbuf_type * std::basic_stringstream< _CharT, _Traits, _Alloc >::rdbuf () const  [inline],
[nodiscard]
```

Accessing the underlying buffer.

Returns

The current basic_stringbuf buffer.

This hides both signatures of std::basic_ios::rdbuf().

rdstate()

```
template<typename _CharT, typename _Traits>
iostate std::basic_ios< _CharT, _Traits >::rdstate () const  [inline], [nodiscard], [inherited]
```

Returns the error state of the stream buffer.

Returns

A bit pattern (well, isn't everything?)

See std::ios_base::iostate for the possible bit values. Most users will call one of the interpreting wrappers, e.g., good().
Referenced by std::basic_ios< char_type, traits_type >::bad(), clear(), std::basic_ios< char_type, traits_type >::eof(), std::basic_ios< char_type, traits_type >::fail(), std::basic_ios< char_type, traits_type >::good(), std::basic_istream< _CharT, _Traits >::p std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ios< char_type, traits_type > and std::basic_istream< _CharT, _Traits >::unget().

read()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::read (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction without delimiters.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

*this

If the stream state is `good()`, extracts characters and stores them into *__s* until one of the following happens:

- *__n* characters are stored
- the input sequence reaches end-of-file, in which case the error state is set to `failbit|eofbit`.

Note

This function is not overloaded on signed char and unsigned char.

References _M_gcount.

readsome()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::readsome (
            char_type * __s,
            streamsize __n)  [inherited]
```

Extraction until the buffer is exhausted, but no more.

Parameters

_↩ _s	A character array.
_↩ _n	Maximum number of characters to store.

Returns

The number of characters extracted.

Extracts characters and stores them into __*s* depending on the number of characters remaining in the streambuf's buffer, `rdbuf()->in_avail()`, called `A` here:

- if `A == -1`, sets eofbit and extracts no characters
- if `A == 0`, extracts no characters
- if `A > 0`, extracts `min(A,n)`

The goal is to empty the current buffer, and to not request any more from the external input sequence controlled by the streambuf.
References _M_gcount, std::ios_base::badbit, std::ios_base::eofbit, std::ios_base::goodbit, std::min(), std::basic_ios< _CharT, _Traits >::
and std::basic_ios< _CharT, _Traits >::setstate().

register_callback()

```
void std::ios_base::register_callback (
             event_callback __fn,
             int __index)  [inherited]
```

Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

seekg() [1/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
             off_type __off,
             ios_base::seekdir __dir)  [inherited]
```

Changing the current read position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(__off,__dir)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_ios< _CharT, _Traits >::setstate().

seekg() [2/2]

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::seekg (
            pos_type __pos)  [inherited]
```

Changing the current read position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(__pos)`. If that function fails, sets failbit.

Note

This function first clears eofbit. It does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::goodbit, std::ios_base::in, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            off_type __off,
            ios_base::seekdir __dir)  [inherited]
```

Changing the current write position.

Parameters

__off	A file offset object.
__dir	The direction in which to seek.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekoff(off,dir)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

seekp() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::seekp (
            pos_type __pos)  [inherited]
```

Changing the current write position.

Parameters

__pos	A file position object.

Returns

*this

If `fail()` is not true, calls `rdbuf()->pubseekpos(pos)`. If that function fails, sets failbit.
References std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::failbit, std::ios_base::out, std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline], [inherited]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* & *mask*. An example mask is `ios_base::adjustfield`.

setstate()

```
template<typename _CharT, typename _Traits>
void std::basic_ios< _CharT, _Traits >::setstate (
            iostate __state)  [inline], [inherited]
```

Sets additional flags in the error state.

Parameters

__state	The additional state flag(s) to set.

See std::ios_base::iostate for the possible bit values.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(),
std::basic_istream< _CharT, _Traits >::get(), std::getline(), std::getline(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ostream< _CharT, _Traits >::operator<<(),
std::operator<<(), std::operator<<(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::ope
std::operator>>(), std::operator>>(), std::operator>>(), std::tr2::operator>>(), std::basic_ostream< _CharT, _Traits >::put(),
std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_istream< _CharT, _Tra
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_ostream< _CharT, _Traits >
std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

str() [1/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
__string_type std::basic_stringstream< _CharT, _Traits, _Alloc >::str () const &  [inline], [nodiscard]
```

Copying out the string buffer.

Returns

```
rdbuf()->str()
```

str() [2/2]

```
template<typename _CharT, typename _Traits, typename _Alloc>
void std::basic_stringstream< _CharT, _Traits, _Alloc >::str (
            const __string_type & __s)  [inline]
```

Setting a new buffer.

Parameters

__s	The string to use as a new sequence.

Calls `rdbuf()->str(s)`.

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_istream< _CharT, _Traits >::sync (
            void )  [inherited]
```

Synchronizing the stream buffer.

Returns

0 on success, -1 on failure

If `rdbuf()` is a null pointer, returns -1.
Otherwise, calls `rdbuf()->pubsync()`, and if that returns -1, sets badbit and returns -1.
Otherwise, returns 0.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`.

References std::ios_base::badbit, std::ios_base::goodbit, std::basic_streambuf< _CharT, _Traits >::pubsync(), std::basic_ios< _CharT, _Traits >::rdbuf(), and std::basic_ios< _CharT, _Traits >::setstate().

sync_with_stdio()

```
bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static], [inherited]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/↩libstdc++/manual/fstreams.html#std.io.filestreams.binary

tellg()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits >::pos_type std::basic_istream< _CharT, _Traits >::tellg (
            void )  [inherited]
```

Getting the current read position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,in)`.

Note

This function does not count the number of characters extracted, if any, and therefore does not affect the next call to `gcount()`. At variance with putback, unget and seekg, eofbit is not cleared first.

References std::ios_base::badbit, std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::in, and std::basic_ios< _CharT, _Traits >::rdbuf().

tellp()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits >::pos_type std::basic_ostream< _CharT, _Traits >::tellp ()  [inherited]
```

Getting the current write position.

Returns

A file position object.

If `fail()` is not false, returns `pos_type`(-1) to indicate failure. Otherwise returns `rdbuf()->pubseekoff(0,cur,out)`.
References std::ios_base::cur, std::basic_ios< _CharT, _Traits >::fail(), std::ios_base::out, and std::basic_ios< _CharT, _Traits >::rdbuf()

tie() [1/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie () const  [inline],
[nodiscard], [inherited]
```

Fetches the current *tied* stream.

Returns

A pointer to the tied stream, or NULL if the stream is not tied.

A stream may be *tied* (or synchronized) to a second output stream. When this stream performs any I/O, the tied stream is first flushed. For example, `std::cin` is tied to `std::cout`.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), and copyfmt().

tie() [2/2]

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > * std::basic_ios< _CharT, _Traits >::tie (
            basic_ostream< _CharT, _Traits > * __tiestr)  [inline], [inherited]
```

Ties this stream to an output stream.

Parameters

__tiestr	The output stream.

Returns

The previously tied output stream, or NULL if the stream was not tied.

This sets up a new tie; see tie() for more.

unget()

```
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & std::basic_istream< _CharT, _Traits >::unget (
            void )  [inherited]
```

Unextracting the previous character.

Returns

*this

If `rdbuf()` is not null, calls `rdbuf()->sungetc(c)`.
If `rdbuf()` is null or if `sungetc()` fails, sets badbit in the error state.

Note

This function first clears eofbit. Since no characters are extracted, the next call to `gcount()` will return 0, as required by DR 60.

References _M_gcount, std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::clear(), std::ios_base::eofbit, std::ios_base::goodbit, std::basic_ios< _CharT, _Traits >::rdbuf(), std::basic_ios< _CharT, _Traits >::rdstate(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_streambuf< _CharT, _Traits >::sungetc().
Referenced by std::__detail::operator>>().

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline], [inherited]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

widen()

```
template<typename _CharT, typename _Traits>
char_type std::basic_ios< _CharT, _Traits >::widen (
            char __c) const  [inline], [inherited]
```

Widens characters.

Parameters

_c	The character to widen.

Returns

The widened character.

Maps a character of `char` to a character of `char_type`.
Returns the result of

```
std::use_facet<ctype<char_type> >(getloc()).widen(c)
```

Additional I10n notes are at https://gcc.gnu.org/onlinedocs/libstdc++/manual/localization.html

Referenced by std::endl(), std::basic_ios< char_type, traits_type >::fill(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::get(), std::basic_istream< char_type, traits_type >::getline(), std::getline(), std::getline(), std::operator<<(), std::operator<<(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F and std::tr2::operator>>().

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard], [inherited]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline], [inherited]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

write()

```
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::basic_ostream< _CharT, _Traits >::write (
            const char_type * __s,
            streamsize __n)  [inherited]
```

Character string insertion.

Parameters

_↩ _s	The array to insert.
_↩ _n	Maximum number of characters to insert.

Returns

*this

Characters are copied from __s and inserted into the stream until one of the following happens:

- __n characters are inserted
- inserting into the output sequence fails (in this case, badbit will be set in the stream's error state)

Note

This function is not overloaded on signed char and unsigned char.

xalloc()

```
int std::ios_base::xalloc () throw ( )   [static], [inherited]
```

Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

6.254.6 Member Data Documentation

_M_gcount

```
template<typename _CharT, typename _Traits>
streamsize std::basic_istream< _CharT, _Traits >::_M_gcount  [protected], [inherited]
```

The number of characters extracted in the previous unformatted function; see gcount().
Referenced by get(), get(), get(), get(), getline(), ignore(), ignore(), ignore(), putback(), read(), readsome(), and unget().

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static], [inherited]
```

A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static], [inherited]
```

Seek to end before each write.
Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate  [static], [inherited]
```

Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit  [static], [inherited]
```

Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_typ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

basefield

```
const fmtflags std::ios_base::basefield  [static], [inherited]
```

A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

```
const seekdir std::ios_base::beg  [static], [inherited]
```

Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

const openmode std::ios_base::binary [static], [inherited]
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static], [inherited]
Insert/extract bool in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

const seekdir std::ios_base::cur [static], [inherited]
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

const fmtflags std::ios_base::dec [static], [inherited]
Converts integer input or generates integer output in decimal base.
Referenced by std::dec(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::opera and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>.

end

const seekdir std::ios_base::end [static], [inherited]
Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

const iostate std::ios_base::eofbit [static], [inherited]
Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:: std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Tra std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, char_traits std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

const iostate std::ios_base::failbit [static], [inherited]
Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char >
std::basic_ofstream< char >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type,
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(),
std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(),
std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_iostream< char_type, traits_type >::seekg(),
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits >
and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed  [static], [inherited]
```
Generate floating-point output in fixed-point notation.
Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield  [static], [inherited]
```
A mask of scientific|fixed. Useful for the 2-arg form of `setf`.
Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit  [static], [inherited]
```
Indicates all is well.
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(),
std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d
std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >:
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(),
std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore()
std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_
std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(),
std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(),
std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::
std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _T
std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, char_traits< _CharT > >::tellg(), std::basic_istream< _Cha
and std::ws().

hex

```
const fmtflags std::ios_base::hex  [static], [inherited]
```
Converts integer input or generates integer output in hexadecimal base.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and
std::hex().

in

```
const openmode std::ios_base::in  [static], [inherited]
```
Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_istringstream< char >::basic_istringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char >
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),

std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::p std::basic_streambuf< char_type, traits_type >::pubseekoff(), std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_streambuf< char_type, traits std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_streambuf< char_type, trait std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< char >::seekpos(), std::basic_filebuf< _CharT, _Traits std::basic_stringbuf< char >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underfl std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static], [inherited]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static], [inherited]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::left(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0

oct

```
const fmtflags std::ios_base::oct  [static], [inherited]
```

Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static], [inherited]
```

Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofs std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::o std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::pubseekoff(), std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_streambuf< char_type, traits_type >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_ostream< _CharT, _Traits std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_streambuf< char_type, traits_type >::seekpos(), std::basic_stringbuf< _Cha std::basic_stringbuf< char >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsput

right

```
const fmtflags std::ios_base::right  [static], [inherited]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).
Referenced by std::right().

scientific

```
const fmtflags std::ios_base::scientific  [static], [inherited]
```

Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

```
const fmtflags std::ios_base::showbase  [static], [inherited]
```
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

```
const fmtflags std::ios_base::showpoint  [static], [inherited]
```
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

```
const fmtflags std::ios_base::showpos  [static], [inherited]
```
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static], [inherited]
```
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x
and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static], [inherited]
```
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static], [inherited]
```
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

```
const fmtflags std::ios_base::uppercase  [static], [inherited]
```
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following file:

- sstream

6.255 std::bernoulli_distribution Class Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef bool result_type

Public Member Functions

- bernoulli_distribution ()
- **bernoulli_distribution** (const param_type &__p)
- bernoulli_distribution (double __p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- double p () const
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const bernoulli_distribution &__d1, const bernoulli_distribution &__d2)

6.255.1 Detailed Description

A Bernoulli random number distribution.
Generates a sequence of true and false values with likelihood p that true will come up and $(1-p)$ that false will appear.

Since

C++11

6.255.2 Member Typedef Documentation

result_type

```
typedef bool std::bernoulli_distribution::result_type
```

The type of the range of the distribution.

6.255.3 Constructor & Destructor Documentation

bernoulli_distribution() [1/2]

```
std::bernoulli_distribution::bernoulli_distribution () [inline]
```

Constructs a Bernoulli distribution with likelihood 0.5.
References bernoulli_distribution().
Referenced by bernoulli_distribution(), and operator==.

bernoulli_distribution() [2/2]

```
std::bernoulli_distribution::bernoulli_distribution (
            double __p)  [inline], [explicit]
```

Constructs a Bernoulli distribution with likelihood p.

Parameters

_↩ _p	[IN] The likelihood of a true result being returned. Must be in the interval $[0, 1]$.

6.255.4 Member Function Documentation

max()

```
result_type std::bernoulli_distribution::max () const  [inline]
```

Returns the least upper bound value of the distribution.
References std::numeric_limits< _Tp >::max().

min()

```
result_type std::bernoulli_distribution::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.
References std::numeric_limits< _Tp >::min().

operator()()

```
template<typename _UniformRandomNumberGenerator>
result_type std::bernoulli_distribution::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
References operator()().
Referenced by operator()().

p()

```
double std::bernoulli_distribution::p () const  [inline]
```

Returns the p parameter of the distribution.

param() [1/2]

```
param_type std::bernoulli_distribution::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
void std::bernoulli_distribution::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
void std::bernoulli_distribution::reset ()  [inline]
```

Resets the distribution state.
Does nothing for a Bernoulli distribution.

6.255.5 Friends And Related Symbol Documentation

operator==

```
bool operator== (
            const bernoulli_distribution & __d1,
            const bernoulli_distribution & __d2)  [friend]
```

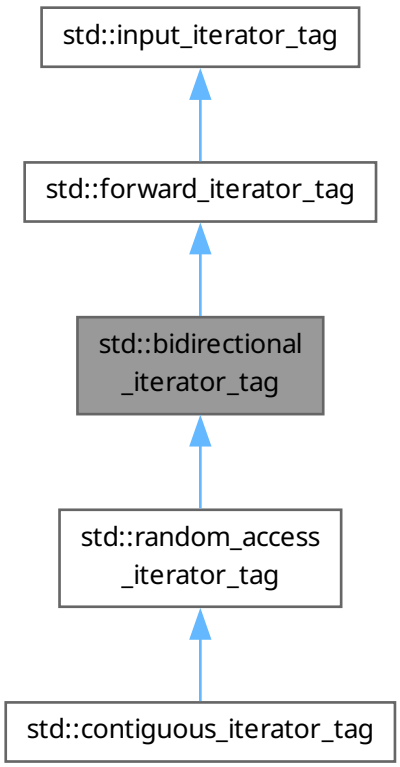
Return true if two Bernoulli distributions have the same parameters.
References bernoulli_distribution().
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.256 std::bidirectional_iterator_tag Struct Reference

```
#include <stl_iterator_base_types.h>
```

Inheritance diagram for std::bidirectional_iterator_tag:



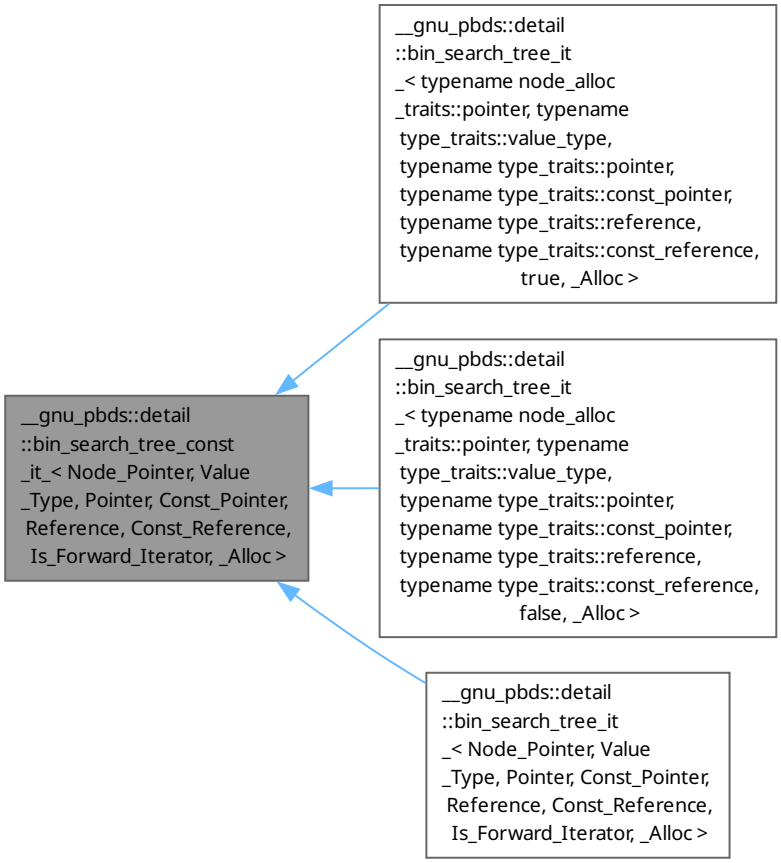
6.256.1 Detailed Description

Bidirectional iterators support a superset of forward iterator operations.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

6.257 __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference

`#include <point_iterators.hpp>`
Inheritance diagram for __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_↩ Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >:



Public Types

- typedef Const_Pointer **const_pointer**
- typedef Const_Reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef std::bidirectional_iterator_tag **iterator_category**

- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value_Type **value_type**

Public Member Functions

- **bin_search_tree_const_it_** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_↩ Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other)
- **bin_search_tree_const_it_** (const Node_Pointer p_nd=0)
- bool **operator!=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other) const
- bool **operator!=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- const_reference **operator*** () const
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator++** ()
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator++** (int)
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator--** ()
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator--** (int)
- const_pointer **operator->** () const
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other)
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)
- bool **operator==** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other) const
- bool **operator==** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const

Public Attributes

- Node_Pointer **m_p_nd**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

6.257.1 Detailed Description

template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename Const_Reference, bool Is_Forward_Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >

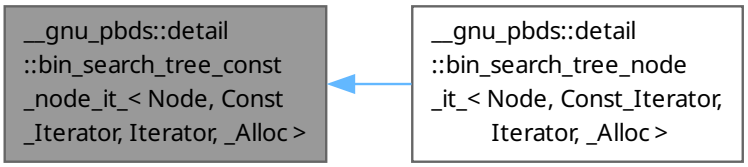
Const iterator.
The documentation for this class was generated from the following file:

- point_iterators.hpp

6.258 __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference

`#include <node_iterators.hpp>`

Inheritance diagram for __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >:



Public Types

- typedef Const_Iterator const_reference
- typedef trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef rebind_traits< _Alloc, metadata_type >::const_reference metadata_const_reference
- typedef Node::metadata_type metadata_type
- typedef Const_Iterator reference
- typedef Const_Iterator value_type

Public Member Functions

- **bin_search_tree_const_node_it_** (const node_pointer p_nd=0)
- bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > get_l_child () const
- metadata_const_reference get_metadata () const
- bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > get_r_child () const
- bool operator!= (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other) const
- const_reference operator* () const
- bool operator== (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

6.258.1 Detailed Description

template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >

Const node iterator.

6.258.2 Member Typedef Documentation

const_reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::const_reference
```

Iterator's __const reference type.

difference_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::difference_type
```

Difference type.

iterator_category

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial_iterator_tag __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_↩
Iterator, Iterator, _Alloc >::iterator_category
```

Category.

metadata_const_reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef rebind_traits<_Alloc,metadata_type>::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference
```

Const metadata reference type.

metadata_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Node::metadata_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_↩
Iterator, Iterator, _Alloc >::metadata_type
```

Metadata type.

reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::reference
```

Iterator's reference type.

value_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Const_Iterator __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::value_type
```

Iterator's value type.

6.258.3 Member Function Documentation

get_l_child()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
```

bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_cons
Node, Const_Iterator, Iterator, _Alloc >::get_l_child () const [inline]
Returns the __const node iterator associated with the left node.

get_metadata()

template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
metadata_const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::get_metadata () const [inline]
Metadata access.

get_r_child()

template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_cons
Node, Const_Iterator, Iterator, _Alloc >::get_r_child () const [inline]
Returns the __const node iterator associated with the right node.

operator"!=()

template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator!= (
const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
other) const [inline]
Compares (negatively) to a different iterator object.

operator∗()

template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator,
_Alloc >::operator* () const [inline]
Access.

operator==()

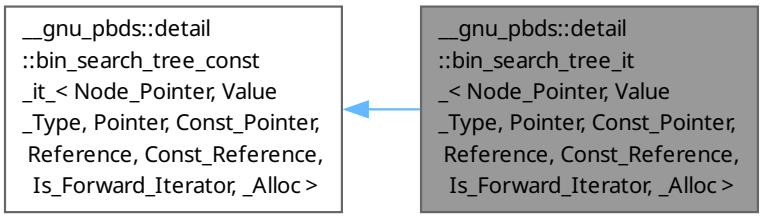
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator== (
const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
other) const [inline]
Compares to a different iterator object.
The documentation for this class was generated from the following file:

- bin_search_tree_/node_iterators.hpp

6.259 __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > Class Template Reference

#include <point_iterators.hpp>

Inheritance diagram for __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >:



Public Types

- typedef Const_Pointer **const_pointer**
- typedef Const_Reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef std::bidirectional_iterator_tag **iterator_category**
- typedef Pointer **pointer**
- typedef Reference **reference**
- typedef Value_Type **value_type**

Public Member Functions

- **bin_search_tree_it_** (const bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other)
- **bin_search_tree_it_** (const Node_Pointer p_nd=0)
- bool **operator!=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other) const
- bool **operator!=** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >::reference **operator*** () const
- bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator++** ()
- bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator++** (int)
- bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator--** ()
- bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > **operator--** (int)
- bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >::pointer **operator->** () const
- bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other)
- bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > & **operator=** (const bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other)

- bool **operator==** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, !Is_Forward_Iterator, _Alloc > &other) const
- bool **operator==** (const bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc > &other) const

Public Attributes

- Node_Pointer **m_p_nd**

Protected Types

- typedef bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_↩ Reference, Is_Forward_Iterator, _Alloc > **base_it_type**

Protected Member Functions

- void **dec** (false_type)
- void **dec** (true_type)
- void **inc** (false_type)
- void **inc** (true_type)

6.259.1 Detailed Description

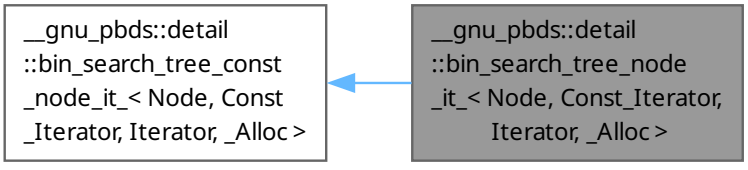
template<typename Node_Pointer, typename Value_Type, typename Pointer, typename Const_Pointer, typename Reference, typename Const_Reference, bool Is_Forward_Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >

Iterator.
The documentation for this class was generated from the following file:

- point_iterators.hpp

6.260 __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > Class Template Reference

`#include <node_iterators.hpp>`
Inheritance diagram for __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >:



Public Types

- typedef Iterator const_reference
- typedef trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef rebind_traits< _Alloc, metadata_type >::const_reference metadata_const_reference
- typedef Node::metadata_type metadata_type
- typedef Iterator reference
- typedef Iterator value_type

Public Member Functions

- **bin_search_tree_node_it_** (const node_pointer p_nd=0)
- bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > get_l_child () const
- metadata_const_reference get_metadata () const
- bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > get_r_child () const
- bool operator!= (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other) const
- Iterator operator* () const
- bool operator== (const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

6.260.1 Detailed Description

template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
class __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >

Node iterator.

6.260.2 Member Typedef Documentation

const_reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _↩
Alloc >::const_reference
```

Iterator's __const reference type.

difference_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial_iterator_difference_type __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::difference_type  [inherited]
```

Difference type.

iterator_category

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef trivial_iterator_tag __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_↩
Iterator, Iterator, _Alloc >::iterator_category  [inherited]
```

Category.

metadata_const_reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef rebind_traits<_Alloc,metadata_type>::const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::metadata_const_reference  [inherited]
```

Const metadata reference type.

metadata_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Node::metadata_type __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_↩
Iterator, Iterator, _Alloc >::metadata_type  [inherited]
```

Metadata type.

reference

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _↩
Alloc >::reference
```

Iterator's reference type.

value_type

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
typedef Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _↩
Alloc >::value_type
```

Iterator's value type.

6.260.3 Member Function Documentation

get_l_child()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::get_l_child () const  [inline]
```

Returns the node iterator associated with the left node.

get_metadata()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
metadata_const_reference __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator,
Iterator, _Alloc >::get_metadata () const  [inline], [inherited]
```

Metadata access.

get_r_child()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc > __gnu_pbds::detail::bin_search_tree_node_it_<
Node, Const_Iterator, Iterator, _Alloc >::get_r_child () const  [inline]
```

Returns the node iterator associated with the right node.

operator"!=()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator!= (
```

```
            const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
other) const  [inline], [inherited]
```

Compares (negatively) to a different iterator object.

operator*()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
Iterator __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator* () const  [inline]
```

Access.

operator==()

```
template<typename Node, class Const_Iterator, class Iterator, typename _Alloc>
bool __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc
>::operator== (
            const bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc > &
other) const  [inline], [inherited]
```

Compares to a different iterator object.

The documentation for this class was generated from the following file:

- bin_search_tree_/node_iterators.hpp

6.261 __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Public Types

- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef Node **node**
- typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_node_it_< Node, point_const_iterator, point_iterator, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node_update_pointer**
- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **reverse_iterator**

6.261.1 Detailed Description

template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >

Binary search tree traits, primary template.

6.261.2 Member Typedef Documentation

node_const_iterator

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_↩
Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::b
Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- bin_search_tree_/traits.hpp

6.262 __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Public Types

- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef Node **node**
- typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef node_const_iterator **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node_update_pointer**
- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef point_const_iterator **point_iterator**
- typedef const_reverse_iterator **reverse_iterator**

6.262.1 Detailed Description

template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class _Cmp_Fn, typename _Alloc > class Node_Update, class Node, typename _Alloc>
struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >

Specialization.

6.262.2 Member Typedef Documentation

node_const_iterator

```
template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class _Cmp_Fn,
typename _Alloc > class Node_Update, class Node, typename _Alloc>
typedef bin_search_tree_const_node_it_< Node, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::b
Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >::node_const_iterator
```

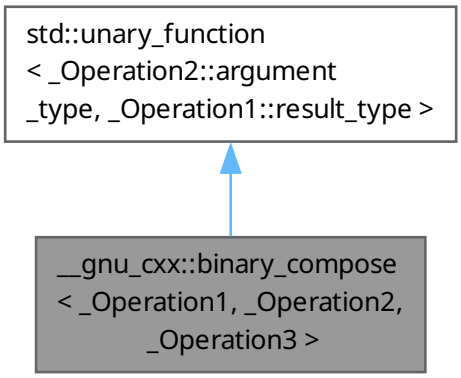
This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- bin_search_tree_/traits.hpp

6.263 __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 > Class Template Reference

`#include <functional>`

Inheritance diagram for __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >:



Public Types

- typedef _Operation2::argument_type argument_type
- typedef _Operation1::result_type result_type

Public Member Functions

- **binary_compose** (const _Operation1 &__x, const _Operation2 &__y, const _Operation3 &__z)
- _Operation1::result_type **operator()** (const typename _Operation2::argument_type &__x) const

Protected Attributes

- _Operation1 **_M_fn1**
- _Operation2 **_M_fn2**
- _Operation3 **_M_fn3**

6.263.1 Detailed Description

template<class _Operation1, class _Operation2, class _Operation3>
class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >

An SGI extension .

6.263.2 Member Typedef Documentation

argument_type

```
typedef _Operation2::argument_type std::unary_function< _Operation2::argument_type, _Operation1::↩
result_type >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _Operation1::result_type std::unary_function< _Operation2::argument_type, _Operation1::↩
result_type >::result_type  [inherited]
```

result_type is the return type

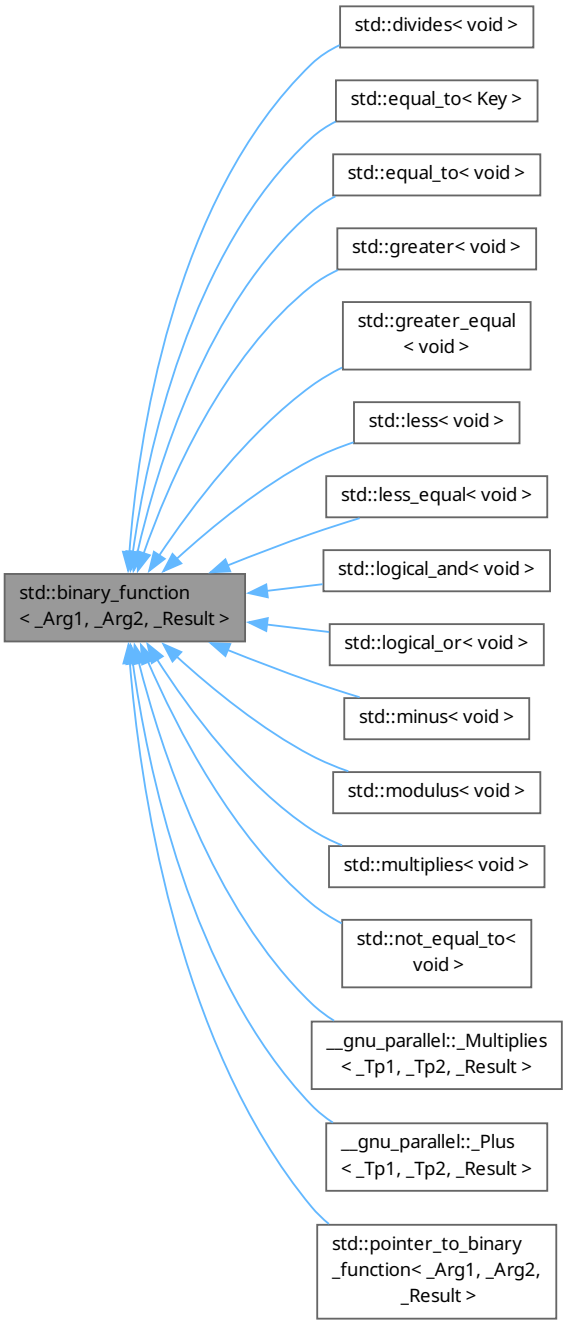
The documentation for this class was generated from the following file:

- ext/functional

6.264 std::binary_function< _Arg1, _Arg2, _Result > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::binary_function< _Arg1, _Arg2, _Result >:



Public Types

- typedef _Arg1 first_argument_type
- typedef _Result result_type
- typedef _Arg2 second_argument_type

6.264.1 Detailed Description

template<typename _Arg1, typename _Arg2, typename _Result>
struct std::binary_function< _Arg1, _Arg2, _Result >

Helper for defining adaptable binary function objects.

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

6.264.2 Member Typedef Documentation

first_argument_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type
```

`first_argument_type` is the type of the first argument

result_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type
```

`result_type` is the return type

second_argument_type

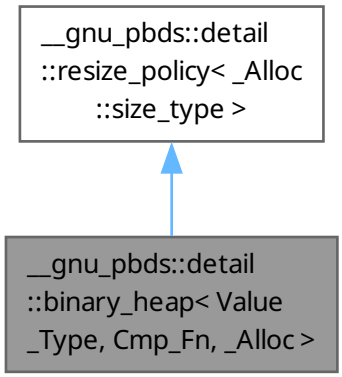
```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.265 __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

`#include <binary_heap_.hpp>`
Inheritance diagram for __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >:


__gnu_pbds::detail
::resize_policy< _Alloc
::size_type >
__gnu_pbds::detail
::binary_heap< Value
_Type, Cmp_Fn, _Alloc >
__gnu_pbds::detail
::binary_heap< Value
_Type, Cmp_Fn, _Alloc >
__gnu_pbds::detail
::resize_policy< _Alloc
::size_type

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef cond_dealtor< value_type, _Alloc > **cond_dealtor_t**
- typedef binary_heap_const_iterator_< value_type, entry, simple_value, _Alloc > **const_iterator**
- typedef __rebind_v::const_pointer **const_pointer**
- typedef __rebind_v::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef __conditional_type< simple_value, value_type, pointer >::__type **entry**
- typedef rebind_traits< _Alloc, entry >::allocator_type **entry_allocator**
- typedef entry_cmp< Value_Type, Cmp_Fn, _Alloc, is_simple< Value_Type >::value >::type **entry_cmp**
- typedef rebind_traits< _Alloc, entry >::pointer **entry_pointer**
- typedef const_iterator **iterator**
- typedef binary_heap_point_const_iterator_< value_type, entry, simple_value, _Alloc > **point_const_iterator**
- typedef point_const_iterator **point_iterator**
- typedef __rebind_v::pointer **pointer**
- typedef __rebind_v::reference **reference**
- typedef __gnu_pbds::detail::resize_policy< typename _Alloc::size_type > **resize_policy**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **binary_heap** (const binary_heap &)
- **binary_heap** (const cmp_fn &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- void **erase** (point_iterator)
- void **erase_at** (entry_pointer, size_type, false_type)
- void **erase_at** (entry_pointer, size_type, true_type)
- template<typename Pred>
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- size_type **get_new_size_for_arbitrary** (size_type) const
- size_type **get_new_size_for_grow** () const
- size_type **get_new_size_for_shrink** () const
- bool **grow_needed** (size_type) const
- void **join** (binary_heap &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **notify_arbitrary** (size_type)
- void **notify_grow_resize** ()
- void **notify_shrink_resize** ()
- void **pop** ()
- point_iterator **push** (const_reference)
- bool **resize_needed_for_grow** (size_type) const

- bool **resize_needed_for_shrink** (size_type) const
- bool **shrink_needed** (size_type) const
- size_type **size** () const
- template<typename Pred>
  void **split** (Pred, binary_heap &)
- void **swap** (binary_heap &)
- void **swap** (resize_policy< _Alloc::size_type > &)
- const_reference **top** () const

Static Public Attributes

- static const _Alloc::size_type **min_size**

Protected Member Functions

- template<typename It>
  void **copy_from_range** (It, It)

6.265.1 Detailed Description

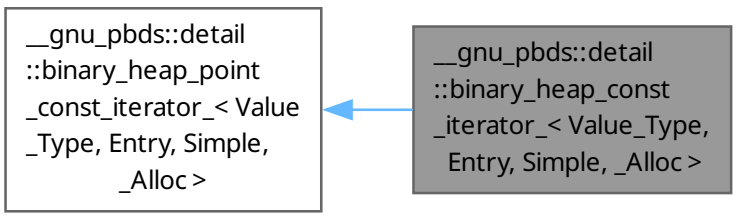
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >

Binary heaps composed of resize and compare policies.
Based on CLRS.
The documentation for this class was generated from the following file:

- binary_heap_.hpp

6.266 __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference

`#include <const_iterator.hpp>`
Inheritance diagram for __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >:



Public Types

- typedef base_type::const_pointer const_pointer
- typedef base_type::const_reference const_reference
- typedef _Alloc::difference_type difference_type

- typedef std::forward_iterator_tag iterator_category
- typedef base_type::pointer pointer
- typedef base_type::reference reference
- typedef base_type::value_type value_type

Public Member Functions

- binary_heap_const_iterator_ ()
- binary_heap_const_iterator_ (const binary_heap_const_iterator_ &other)
- **binary_heap_const_iterator_** (entry_pointer p_e)
- bool operator!= (const binary_heap_const_iterator_ &other) const
- bool operator!= (const binary_heap_point_const_iterator_ &other) const
- const_reference operator* () const
- binary_heap_const_iterator_ & **operator++** ()
- binary_heap_const_iterator_ **operator++** (int)
- const_pointer operator-> () const
- bool operator== (const binary_heap_const_iterator_ &other) const
- bool operator== (const binary_heap_point_const_iterator_ &other) const

Public Attributes

- entry_pointer **m_p_e**

6.266.1 Detailed Description

template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >

Const point-type iterator.

6.266.2 Member Typedef Documentation

const_pointer

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::const_pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::const_pointer
```

Iterator's const pointer type.

const_reference

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::const_reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::const_reference
```

Iterator's const reference type.

difference_type

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef _Alloc::difference_type __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::difference_type
```

Difference type.

iterator_category

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef std::forward_iterator_tag __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::iterator_category
```
Category.

pointer

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::pointer __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::pointer
```
Iterator's pointer type.

reference

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::reference __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::reference
```
Iterator's reference type.

value_type

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef base_type::value_type __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::value_type
```
Iterator's value type.

6.266.3 Constructor & Destructor Documentation

binary_heap_const_iterator_() [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_↩
heap_const_iterator_ ()  [inline]
```
Default constructor.

binary_heap_const_iterator_() [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::binary_↩
heap_const_iterator_ (
            const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
[inline]
```
Copy constructor.

6.266.4 Member Function Documentation

operator"!=() [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::↩
operator!= (
            const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const  [inline]
```
Compares content (negatively) to a different iterator object.

operator"!=() [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc
>::operator!= (
            const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const  [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

operator*()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple,
_Alloc >::operator* () const  [inline], [inherited]
```

Access.

operator->()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, ↩
_Alloc >::operator-> () const  [inline], [inherited]
```

Access.

operator==() [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >::↩
operator== (
            const binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const  [inline]
```

Compares content to a different iterator object.

operator==() [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc
>::operator== (
            const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const  [inline], [inherited]
```

Compares content to a different iterator object.

The documentation for this class was generated from the following file:

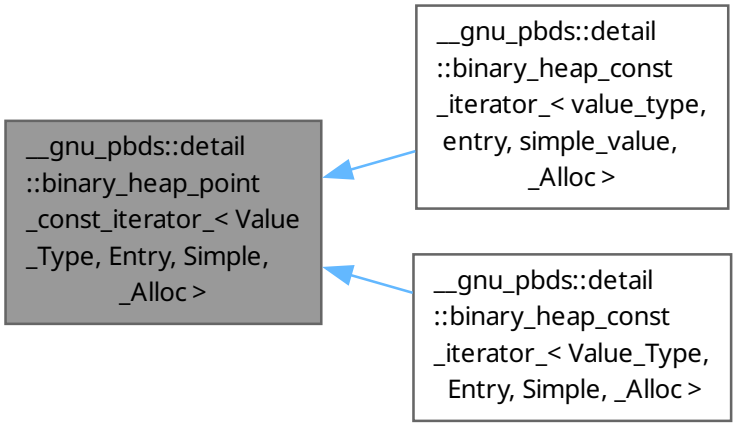
- binary_heap_/const_iterator.hpp

6.267 __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > Class Template Reference

```
#include <point_const_iterator.hpp>
```

Inheritance diagram for __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc

>:



Public Types

- typedef rebind_traits< _Alloc, value_type >::const_pointer const_pointer
- typedef rebind_traits< _Alloc, value_type >::const_reference const_reference
- typedef trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef rebind_traits< _Alloc, value_type >::pointer pointer
- typedef rebind_traits< _Alloc, value_type >::reference reference
- typedef Value_Type value_type

Public Member Functions

- binary_heap_point_const_iterator_ ()
- binary_heap_point_const_iterator_ (const binary_heap_point_const_iterator_ &other)
- **binary_heap_point_const_iterator_** (entry_pointer p_e)
- bool operator!= (const binary_heap_point_const_iterator_ &other) const
- const_reference operator* () const
- const_pointer operator-> () const
- bool operator== (const binary_heap_point_const_iterator_ &other) const

Public Attributes

- entry_pointer **m_p_e**

Protected Types

- typedef rebind_traits< _Alloc, Entry >::pointer **entry_pointer**

6.267.1 Detailed Description

template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
class __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >

Const point-type iterator.

6.267.2 Member Typedef Documentation

const_pointer

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::const_pointer
```

Iterator's const pointer type.

const_reference

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::const_reference
```

Iterator's const reference type.

difference_type

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef trivial_iterator_difference_type __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::difference_type
```

Difference type.

iterator_category

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef trivial_iterator_tag __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type,
Entry, Simple, _Alloc >::iterator_category
```

Category.

pointer

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::pointer __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::pointer
```

Iterator's pointer type.

reference

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::reference __gnu_pbds::detail::binary_heap_point_const_iterator_<
Value_Type, Entry, Simple, _Alloc >::reference
```

Iterator's reference type.

value_type

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
typedef Value_Type __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry,
Simple, _Alloc >::value_type
```

Iterator's value type.

6.267.3 Constructor & Destructor Documentation

binary_heap_point_const_iterator_() [1/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::↩
binary_heap_point_const_iterator_ ()  [inline]
```

Default constructor.

binary_heap_point_const_iterator_() [2/2]

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >::↩
binary_heap_point_const_iterator_ (
            const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
[inline]
```

Copy constructor.

6.267.4 Member Function Documentation

operator"!=()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc
>::operator!= (
            const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const  [inline]
```

Compares content (negatively) to a different iterator object.

operator*()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_reference __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple,
_Alloc >::operator* () const  [inline]
```

Access.

operator->()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
const_pointer __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, ↩
_Alloc >::operator-> () const  [inline]
```

Access.

operator==()

```
template<typename Value_Type, typename Entry, bool Simple, typename _Alloc>
bool __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc
>::operator== (
            const binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc > & other)
const  [inline]
```

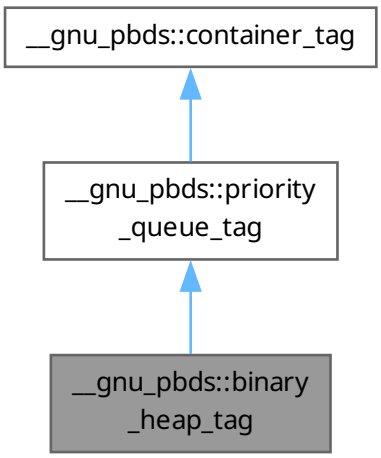
Compares content to a different iterator object.

The documentation for this class was generated from the following file:

- binary_heap_/point_const_iterator.hpp

6.268 __gnu_pbds::binary_heap_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::binary_heap_tag:



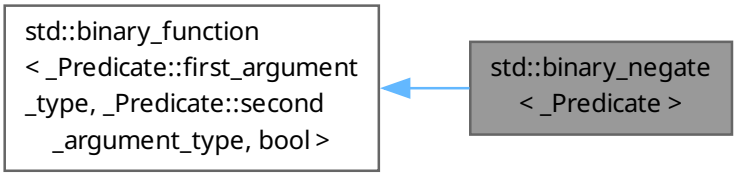
6.268.1 Detailed Description

Binary-heap (array-based).
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.269 std::binary_negate< _Predicate > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::binary_negate< _Predicate >:



Public Types

- typedef _Predicate::first_argument_type first_argument_type

- typedef bool result_type
- typedef _Predicate::second_argument_type second_argument_type

Public Member Functions

- constexpr **binary_negate** (const _Predicate &__x)
- constexpr bool **operator()** (const typename _Predicate::first_argument_type &__x, const typename _↩ Predicate::second_argument_type &__y) const

Protected Attributes

- _Predicate **_M_pred**

6.269.1 Detailed Description

template<typename _Predicate>
class std::binary_negate< _Predicate >

One of the negation functors.

6.269.2 Member Typedef Documentation

first_argument_type

```
typedef _Predicate::first_argument_type std::binary_function< _Predicate::first_argument_type, _↩
Predicate::second_argument_type, bool >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< _Predicate::first_argument_type, _Predicate::second_argument_↩
type, bool >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _Predicate::second_argument_type std::binary_function< _Predicate::first_argument_type, ↩
_Predicate::second_argument_type, bool >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument

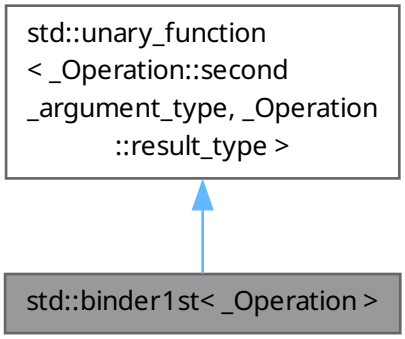
The documentation for this class was generated from the following file:

- stl_function.h

6.270 std::binder1st< _Operation > Class Template Reference

```
#include <binders.h>
```

Inheritance diagram for std::binder1st< _Operation >:



Public Types

- typedef _Operation::second_argument_type argument_type
- typedef _Operation::result_type result_type

Public Member Functions

- **binder1st** (const _Operation &__x, const typename _Operation::first_argument_type &__y)
- _Operation::result_type **operator()** (const typename _Operation::second_argument_type &__x) const
- _Operation::result_type **operator()** (typename _Operation::second_argument_type &__x) const

Protected Attributes

- _Operation **op**
- _Operation::first_argument_type **value**

6.270.1 Detailed Description

template<typename _Operation>
class std::binder1st< _Operation >

One of the binder functors.

6.270.2 Member Typedef Documentation

argument_type

```
typedef _Operation::second_argument_type std::unary_function< _Operation::second_argument_type, ↩
_Operation::result_type >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _Operation::result_type std::unary_function< _Operation::second_argument_type, _Operation::↩
result_type >::result_type  [inherited]
```

result_type is the return type

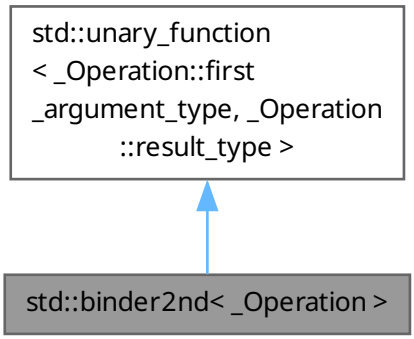
The documentation for this class was generated from the following file:

- backward/binders.h

6.271 std::binder2nd< _Operation > Class Template Reference

```
#include <binders.h>
```

Inheritance diagram for std::binder2nd< _Operation >:



Public Types

- typedef _Operation::first_argument_type argument_type
- typedef _Operation::result_type result_type

Public Member Functions

- **binder2nd** (const _Operation &__x, const typename _Operation::second_argument_type &__y)
- _Operation::result_type **operator()** (const typename _Operation::first_argument_type &__x) const
- _Operation::result_type **operator()** (typename _Operation::first_argument_type &__x) const

Protected Attributes

- _Operation **op**
- _Operation::second_argument_type **value**

6.271.1 Detailed Description

template<typename _Operation>
class std::binder2nd< _Operation >

One of the binder functors.

6.271.2 Member Typedef Documentation

argument_type

```
typedef _Operation::first_argument_type std::unary_function< _Operation::first_argument_type, _↩
Operation::result_type >::argument_type  [inherited]
```

argument_type is the type of the argument

result_type

```
typedef _Operation::result_type std::unary_function< _Operation::first_argument_type, _Operation::↩
result_type >::result_type  [inherited]
```

result_type is the return type

The documentation for this class was generated from the following file:

- backward/binders.h

6.272 std::binomial_distribution< _IntType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- **binomial_distribution** (_IntType __t, double __p=0.5)
- **binomial_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- double p () const
- param_type param () const
- void param (const param_type &__param)
- void reset ()
- _IntType t () const

Friends

- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::binomial_distribution< _IntType1 > &__x)
- bool operator== (const binomial_distribution &__d1, const binomial_distribution &__d2)
- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::binomial_distribution< _IntType1 > &__x)

6.272.1 Detailed Description

template<typename _IntType = int>
class std::binomial_distribution< _IntType >

A discrete binomial random number distribution.
The formula for the binomial probability density function is $p(i|t,p) = \binom{t}{i} p^i (1-p)^{t-i}$ where t and p are the parameters of the distribution.

Since

C++11

6.272.2 Member Typedef Documentation

result_type

```
template<typename _IntType = int>
typedef _IntType std::binomial_distribution< _IntType >::result_type
```

The type of the range of the distribution.

6.272.3 Member Function Documentation

max()

```
template<typename _IntType = int>
result_type std::binomial_distribution< _IntType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _IntType = int>
result_type std::binomial_distribution< _IntType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()() [1/2]

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::binomial_distribution< _IntType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

operator()() [2/2]

```
template<typename _IntType>
template<typename _UniformRandomNumberGenerator>
binomial_distribution< _IntType >::result_type std::binomial_distribution< _IntType >::operator()
(
            _UniformRandomNumberGenerator & __urng,
            const param_type & __param)
```

A rejection algorithm when t * p >= 8 and a simple waiting time method - the second in the referenced book - otherwise. NB: The former is available only if _GLIBCXX_USE_C99_MATH_FUNCS is defined.
Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sect. 4 (+ Errata!).
References std::abs(), std::numeric_limits< _Tp >::epsilon(), std::log(), and std::numeric_limits< _Tp >::max().

p()

```
template<typename _IntType = int>
double std::binomial_distribution< _IntType >::p () const  [inline]
```

Returns the distribution p parameter.

param() [1/2]

```
template<typename _IntType = int>
param_type std::binomial_distribution< _IntType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _IntType = int>
void std::binomial_distribution< _IntType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _IntType = int>
void std::binomial_distribution< _IntType >::reset ()  [inline]
```

Resets the distribution state.

t()

```
template<typename _IntType = int>
_IntType std::binomial_distribution< _IntType >::t () const  [inline]
```

Returns the distribution t parameter.

6.272.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
```

```
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::binomial_distribution< _IntType1 > & __x)  [friend]
```

Inserts a binomial_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A binomial_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _IntType = int>
bool operator== (
            const binomial_distribution< _IntType > & __d1,
            const binomial_distribution< _IntType > & __d2)  [friend]
```

Return true if two binomial distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::binomial_distribution< _IntType1 > & __x)  [friend]
```

Extracts a binomial_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A binomial_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

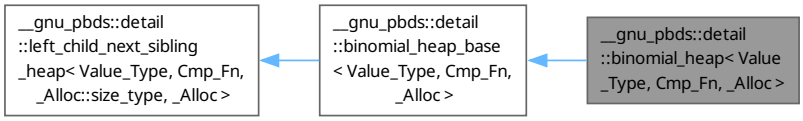
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.273 __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

```
#include <binomial_heap_.hpp>
```

Inheritance diagram for __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef base_type::allocator_type **allocator_type**
- typedef base_type::cmp_fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **binomial_heap** (const binomial_heap &)
- **binomial_heap** (const Cmp_Fn &)
- iterator **begin** ()
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- void **erase** (point_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- void **join** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred>
  void **split** (Pred, binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef alloc_traits::allocator_type **node_allocator**
- typedef _Alloc::size_type **node_metadata**
- typedef std::pair< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It>
  void **copy_from_range** (It, It)
- void **find_max** ()
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

6.273.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >

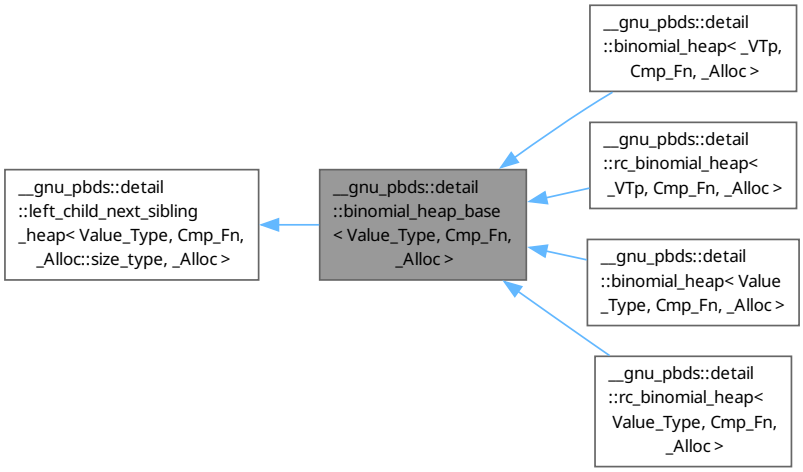
Binomial heap.
The documentation for this class was generated from the following file:

- binomial_heap_.hpp

6.274 __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

`#include <binomial_heap_base_.hpp>`

Inheritance diagram for __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef __rebind_v::const_pointer **const_pointer**
- typedef __rebind_v::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef __rebind_v::pointer **pointer**
- typedef __rebind_v::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- iterator **begin** ()
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- void **erase** (point_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- void **join** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()

- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred>
  void **split** (Pred, binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef alloc_traits::allocator_type **node_allocator**
- typedef base_type::node_const_pointer **node_const_pointer**
- typedef _Alloc::size_type **node_metadata**
- typedef base_type::node_pointer **node_pointer**
- typedef std::pair< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- **binomial_heap_base** (const binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- **binomial_heap_base** (const Cmp_Fn &)
- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It>
  void **copy_from_range** (It, It)
- void **find_max** ()
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

6.274.1 Detailed Description

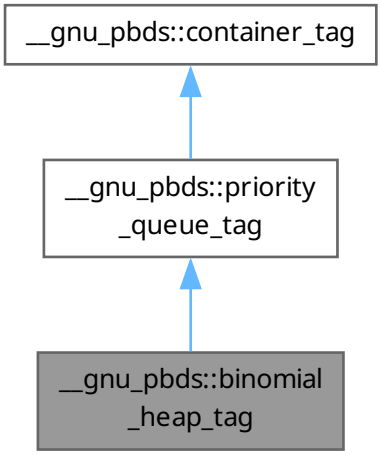
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >

Base class for binomial heap.
The documentation for this class was generated from the following file:

- binomial_heap_base_.hpp

6.275 __gnu_pbds::binomial_heap_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::binomial_heap_tag:



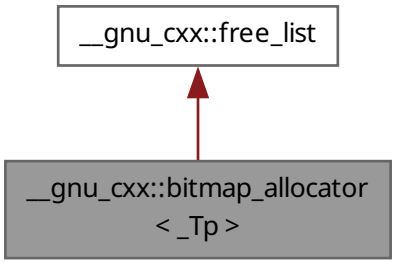
6.275.1 Detailed Description

Binomial-heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.276 __gnu_cxx::bitmap_allocator< _Tp > Class Template Reference

`#include <bitmap_allocator.h>`
Inheritance diagram for __gnu_cxx::bitmap_allocator< _Tp >:



Public Types

- typedef free_list::__mutex_type **__mutex_type**
- typedef const _Tp * **const_pointer**
- typedef const _Tp & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Tp * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef _Tp & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **bitmap_allocator** (const bitmap_allocator &) noexcept
- template<typename _Tp1>
  bitmap_allocator (const bitmap_allocator< _Tp1 > &) noexcept
- pointer _M_allocate_single_object ()
- void _M_deallocate_single_object (pointer __p) throw ()
- const_pointer **address** (const_reference __r) const noexcept
- pointer **address** (reference __r) const noexcept
- pointer **allocate** (size_type __n)
- pointer **allocate** (size_type __n, typename bitmap_allocator< void >::const_pointer)
- template<typename _Up, typename... _Args>
  void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n) throw ()
- template<typename _Up>
  void **destroy** (_Up *__p)
- size_type **max_size** () const noexcept

6.276.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::bitmap_allocator< _Tp >

Bitmap Allocator, primary template.

6.276.2 Member Function Documentation

_M_allocate_single_object()

```
template<typename _Tp>
pointer __gnu_cxx::bitmap_allocator< _Tp >::_M_allocate_single_object ()  [inline]
```

Allocates memory for a single object of size sizeof(_Tp).

Exceptions

std::bad_alloc.	If memory cannot be allocated.

Complexity: Worst case complexity is O(N), but that is hardly ever hit. If and when this particular case is encountered, the next few cases are guaranteed to have a worst case complexity of O(1)! That's why this function performs very well on average. You can consider this function to have a complexity referred to commonly as: Amortized Constant time.

_M_deallocate_single_object()

```
template<typename _Tp>
void __gnu_cxx::bitmap_allocator< _Tp >::_M_deallocate_single_object (
            pointer __p) throw ( )   [inline]
```

Deallocates memory that belongs to a single object of size sizeof(_Tp).
Complexity: O(lg(N)), but the worst case is not hit often! This is because containers usually deallocate memory close to each other and this case is handled in O(1) time by the deallocate function.
The documentation for this class was generated from the following file:

- bitmap_allocator.h

6.277 std::__debug::bitset< _Nb > Class Template Reference

```
#include <bitset>
```

Public Types

- typedef _Base::reference **reference**

Public Member Functions

- constexpr **bitset** (const _Base &__x)
- template<typename _CharT, typename = _Require<is_trivially_copyable<_CharT>, is_standard_layout<_CharT>, is_trivially_default_↩ constructible<_CharT>, __not_<is_array<_CharT>>>>
  constexpr **bitset** (const _CharT *__str, typename std::basic_string< _CharT >::size_type __n=std::basic_string< _CharT >::npos, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'))
- template<class _CharT, class _Traits, class _Alloc>
  constexpr **bitset** (const std::basic_string< _CharT, _Traits, _Alloc > &__str, typename std::basic_string< _↩ CharT, _Traits, _Alloc >::size_type __pos, typename std::basic_string< _CharT, _Traits, _Alloc >::size_type __n, _CharT __zero, _CharT __one=_CharT('1'))
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr **bitset** (const std::basic_string< _CharT, _Traits, _Alloc > &__str, typename std::basic_string< _↩ CharT, _Traits, _Alloc >::size_type __pos=0, typename std::basic_string< _CharT, _Traits, _Alloc >::size_type __n=(std::basic_string< _CharT, _Traits, _Alloc >::npos))
- constexpr **bitset** (unsigned long long __val) noexcept
- constexpr const _Base & **_M_base** () const noexcept
- constexpr _Base & **_M_base** () noexcept
- constexpr bitset< _Nb > & **flip** () noexcept
- constexpr bitset< _Nb > & **flip** (size_t __pos)
- constexpr bitset< _Nb > & **operator&=** (const bitset< _Nb > &__rhs) noexcept
- constexpr bitset< _Nb > **operator**<< (size_t __pos) const noexcept
- constexpr bitset< _Nb > & **operator**<<**=** (size_t __pos) noexcept
- constexpr bool **operator==** (const bitset< _Nb > &__rhs) const noexcept
- constexpr bitset< _Nb > **operator**>> (size_t __pos) const noexcept
- constexpr bitset< _Nb > & **operator**>>**=** (size_t __pos) noexcept
- constexpr reference **operator[ ]** (size_t __pos)
- constexpr bool **operator[ ]** (size_t __pos) const
- constexpr bitset< _Nb > & **operator^=** (const bitset< _Nb > &__rhs) noexcept
- constexpr bitset< _Nb > & **operator|=** (const bitset< _Nb > &__rhs) noexcept
- constexpr bitset< _Nb > **operator~** () const noexcept
- constexpr bitset< _Nb > & **reset** () noexcept
- constexpr bitset< _Nb > & **reset** (size_t __pos)
- constexpr bitset< _Nb > & **set** () noexcept

- constexpr bitset< _Nb > & **set** (size_t __pos, bool __val=true)
- constexpr std::basic_string< char, std::char_traits< char >, std::allocator< char > > **to_string** () const
- template<typename _CharT>
  constexpr std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > > **to_string** () const
- template<typename _CharT, typename _Traits>
  constexpr std::basic_string< _CharT, _Traits, std::allocator< _CharT > > **to_string** () const
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr std::basic_string< _CharT, _Traits, _Alloc > **to_string** () const
- template<class _CharT>
  constexpr std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT, class _Traits>
  constexpr std::basic_string< _CharT, _Traits, std::allocator< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT, class _Traits, class _Alloc>
  constexpr std::basic_string< _CharT, _Traits, _Alloc > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- constexpr std::basic_string< char, std::char_traits< char >, std::allocator< char > > **to_string** (char __zero, char __one='1') const

6.277.1 Detailed Description

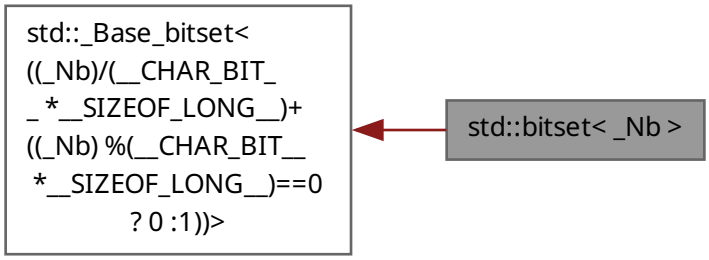
template<size_t _Nb>
class std::__debug::bitset< _Nb >

Class std::bitset with additional safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/bitset

6.278 std::bitset< _Nb > Class Template Reference

`#include <bitset>`
Inheritance diagram for std::bitset< _Nb >:



Classes

- class reference

Public Member Functions

- constexpr bitset () noexcept
- template<typename _CharT, typename = _Require<is_trivially_copyable<_CharT>, is_standard_layout<_CharT>, is_trivially_default_↩ constructible<_CharT>, __not_<is_array<_CharT>>>>
  constexpr bitset (const _CharT *__str, typename __bitset::__string< _CharT >::size_type __n=__bitset::__↩ string< _CharT >::npos, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'))
- template<class _CharT, class _Traits, class _Alloc>
  constexpr bitset (const std::basic_string< _CharT, _Traits, _Alloc > &__s, size_t __position, size_t __n)
- template<class _CharT, class _Traits, class _Alloc>
  constexpr **bitset** (const std::basic_string< _CharT, _Traits, _Alloc > &__s, size_t __position, size_t __n, _CharT __zero, _CharT __one=_CharT('1'))
- template<class _CharT, class _Traits, class _Alloc>
  constexpr bitset (const std::basic_string< _CharT, _Traits, _Alloc > &__s, size_t __position=0)
- constexpr bitset (unsigned long long __val) noexcept
- constexpr size_t _Find_first () const noexcept
- constexpr size_t _Find_next (size_t __prev) const noexcept
- constexpr bool all () const noexcept
- constexpr bool any () const noexcept
- constexpr size_t count () const noexcept
- constexpr bitset< _Nb > & flip () noexcept
- constexpr bitset< _Nb > & flip (size_t __position)
- constexpr bool none () const noexcept
- constexpr bitset< _Nb > operator~ () const noexcept
- constexpr bitset< _Nb > & reset () noexcept
- constexpr bitset< _Nb > & reset (size_t __position)
- constexpr bitset< _Nb > & set () noexcept
- constexpr bitset< _Nb > & set (size_t __position, bool __val=true)
- constexpr size_t size () const noexcept
- constexpr bool test (size_t __position) const
- constexpr std::basic_string< char, std::char_traits< char >, std::allocator< char > > **to_string** () const
- template<class _CharT>
  constexpr std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > > **to_string** () const
- template<class _CharT, class _Traits>
  constexpr std::basic_string< _CharT, _Traits, std::allocator< _CharT > > **to_string** () const
- template<class _CharT, class _Traits, class _Alloc>
  constexpr std::basic_string< _CharT, _Traits, _Alloc > to_string () const
- template<class _CharT>
  constexpr std::basic_string< _CharT, std::char_traits< _CharT >, std::allocator< _CharT > > **to_string** (_↩ CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT, class _Traits>
  constexpr std::basic_string< _CharT, _Traits, std::allocator< _CharT > > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- template<class _CharT, class _Traits, class _Alloc>
  constexpr std::basic_string< _CharT, _Traits, _Alloc > **to_string** (_CharT __zero, _CharT __one=_CharT('1')) const
- constexpr std::basic_string< char, std::char_traits< char >, std::allocator< char > > **to_string** (char __zero, char __one='1') const
- constexpr unsigned long long **to_ullong** () const
- constexpr unsigned long to_ulong () const

- constexpr bitset< _Nb > & operator&= (const bitset< _Nb > &__rhs) noexcept

- constexpr bitset< _Nb > & operator|= (const bitset< _Nb > &__rhs) noexcept
- constexpr bitset< _Nb > & operator^= (const bitset< _Nb > &__rhs) noexcept

- constexpr bitset< _Nb > & operator<<= (size_t __position) noexcept
- constexpr bitset< _Nb > & operator>>= (size_t __position) noexcept

- constexpr bitset< _Nb > & _Unchecked_set (size_t __pos) noexcept
- constexpr bitset< _Nb > & _Unchecked_set (size_t __pos, int __val) noexcept
- constexpr bitset< _Nb > & _Unchecked_reset (size_t __pos) noexcept
- constexpr bitset< _Nb > & _Unchecked_flip (size_t __pos) noexcept
- constexpr bool _Unchecked_test (size_t __pos) const noexcept

- constexpr reference operator[ ] (size_t __position)
- constexpr bool operator[ ] (size_t __position) const

- constexpr bool operator== (const bitset< _Nb > &__rhs) const noexcept

- constexpr bitset< _Nb > operator<< (size_t __position) const noexcept
- constexpr bitset< _Nb > operator>> (size_t __position) const noexcept

Friends

- template<class _CharT, class _Traits, size_t _Nb2>
  std::basic_ostream< _CharT, _Traits > & **operator**<< (std::basic_ostream< _CharT, _Traits > &, const bitset< _Nb2 > &)
- template<class _CharT, class _Traits, size_t _Nb2>
  std::basic_istream< _CharT, _Traits > & **operator**>> (std::basic_istream< _CharT, _Traits > &, bitset< _Nb2 > &)
- class **reference**
- struct **std::hash**< **bitset** >

6.278.1 Detailed Description

template<**size_t _Nb**>
class std::bitset< **_Nb** >

The bitset class represents a *fixed-size* sequence of bits.
(Note that bitset does *not* meet the formal requirements of a `container`. Mainly, it lacks iterators.)
The template argument, `Nb`, may be any non-negative number, specifying the number of bits (e.g., "0", "12", "1024*1024").
In the general unoptimized case, storage is allocated in word-sized blocks. Let B be the number of bits in a word, then (Nb+(B-1))/B words will be used for storage. B - NbB bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.
If you think of bitset as *a simple array of bits*, be aware that your mental picture is reversed: a bitset behaves the same way as bits in integers do, with the bit at index 0 in the *least significant / right-hand* position, and the bit at index Nb-1 in the *most significant / left-hand* position. Thus, unlike other containers, a bitset's index *counts from right to left*, to put it very loosely.

This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints *b('a') is 0001100001* on a modern ASCII system.

```
#include <bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
    long         a = 'a';
    bitset<10>   b(a);

    cout « "b('a') is " « b « endl;

    ostringstream s;
    s « b;
    string  str = s.str();
    cout « "index 3 in the string is " « str[3] « " but\n"
         « "index 3 in the bitset is " « b[3] « endl;
}
```

Also see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext_containers.html for a description of extensions.

Most of the actual code isn't contained in bitset<> itself, but in the base class _Base_bitset. The base class works with whole words, not with individual bits. This allows us to specialize _Base_bitset for the important special case where the bitset is only a single word.

Extra confusion can result due to the fact that the storage for _Base_bitset *is* a regular array, and is indexed as such. This is carefully encapsulated.

6.278.2 Constructor & Destructor Documentation

bitset() [1/5]

```
template<size_t _Nb>
std::bitset< _Nb >::bitset ()  [inline], [constexpr], [noexcept]
```

All bits set to zero.

Referenced by operator&=(), operator<<(), operator<<=(), operator==(), operator>>(), operator^=(), operator|=(), and operator~().

bitset() [2/5]

```
template<size_t _Nb>
std::bitset< _Nb >::bitset (
            unsigned long long __val)  [inline], [constexpr], [noexcept]
```

Initial bits bitwise-copied from a single word (others set to zero).

bitset() [3/5]

```
template<size_t _Nb>
template<class _CharT, class _Traits, class _Alloc>
std::bitset< _Nb >::bitset (
            const std::basic_string< _CharT, _Traits, _Alloc > & __s,
            size_t __position = 0)  [inline], [explicit], [constexpr]
```

Use a subset of a string.

Parameters

__s	A string of 0 and 1 characters.
__position	Index of the first character in __s to use; defaults to zero.

Exceptions

std::out_of_range	If __position > __s.size().
std::invalid_argument	If a character appears in the string which is neither 0 nor 1.

References std::basic_string< _CharT, _Traits, _Alloc >::npos.

bitset() [4/5]

```
template<size_t _Nb>
template<class _CharT, class _Traits, class _Alloc>
std::bitset< _Nb >::bitset (
            const std::basic_string< _CharT, _Traits, _Alloc > & __s,
            size_t __position,
            size_t __n)  [inline], [constexpr]
```

Use a subset of a string.

Parameters

__s	A string of 0 and 1 characters.
__position	Index of the first character in __s to use.
__n	The number of characters to copy.

Exceptions

std::out_of_range	If __position > __s.size().
std::invalid_argument	If a character appears in the string which is neither 0 nor 1.

bitset() [5/5]

```
template<size_t _Nb>
template<typename _CharT, typename = _Require<is_trivially_copyable<_CharT>, is_standard_↩
layout<_CharT>, is_trivially_default_constructible<_CharT>, __not_<is_array<_CharT>>>>>
std::bitset< _Nb >::bitset (
            const _CharT * __str,
            typename __bitset< _Nb >::__string< _CharT >::size_type __n = __bitset< _Nb >::__↩
string<_CharT>::npos,
            _CharT __zero = _CharT('0'),
            _CharT __one = _CharT('1'))  [inline], [explicit], [constexpr]
```

Construct from a character array.

Parameters

__str	An array of characters __zero and __one.
__n	The number of characters to use.

__zero	The character corresponding to the value 0.
__one	The character corresponding to the value 1.

Exceptions

std::invalid_argument	If a character appears in the string which is neither __zero nor __one.

References std::basic_string< _CharT >::npos.

6.278.3 Member Function Documentation

all()

```
template<size_t _Nb>
bool std::bitset< _Nb >::all () const  [inline], [constexpr], [noexcept]
```

Tests whether all the bits are on.

Returns

True if all the bits are set.

any()

```
template<size_t _Nb>
bool std::bitset< _Nb >::any () const  [inline], [constexpr], [noexcept]
```

Tests whether any of the bits are on.

Returns

True if at least one bit is set.

count()

```
template<size_t _Nb>
size_t std::bitset< _Nb >::count () const  [inline], [constexpr], [noexcept]
```

Returns the number of bits which are set.

flip() [1/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::flip ()  [inline], [constexpr], [noexcept]
```

Toggles every bit to its opposite value.

flip() [2/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::flip (
            size_t __position)  [inline], [constexpr]
```

Toggles a given bit to its opposite value.

Parameters

__position	The index of the bit.

Exceptions

std::out_of_range	If *pos* is bigger the size of the set.

References _Unchecked_flip().

none()

```
template<size_t _Nb>
bool std::bitset< _Nb >::none () const  [inline], [constexpr], [noexcept]
```

Tests whether any of the bits are on.

Returns

True if none of the bits are set.

operator&=()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator&= (
            const bitset< _Nb > & __rhs)  [inline], [constexpr], [noexcept]
```

Operations on bitsets.

Parameters

__rhs	A same-sized bitset.

These should be self-explanatory.
References bitset().

operator<<()

```
template<size_t _Nb>
bitset< _Nb > std::bitset< _Nb >::operator<< (
            size_t __position) const  [inline], [constexpr], [noexcept]
```

Self-explanatory.
References bitset().

operator<<=()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator<<= (
            size_t __position)  [inline], [constexpr], [noexcept]
```

Operations on bitsets.

Parameters

__position	The number of places to shift.

These should be self-explanatory.
References bitset(), and operator<<=().
Referenced by operator<<=().

operator==()

```
template<size_t _Nb>
bool std::bitset< _Nb >::operator== (
            const bitset< _Nb > & __rhs) const  [inline], [constexpr], [noexcept]
```

These comparisons for equality/inequality are, well, *bitwise*.
References bitset().

operator>>()

```
template<size_t _Nb>
bitset< _Nb > std::bitset< _Nb >::operator>> (
            size_t __position) const  [inline], [constexpr], [noexcept]
```

Self-explanatory.
References bitset().

operator>>=()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator>>= (
            size_t __position)  [inline], [constexpr], [noexcept]
```

Operations on bitsets.

Parameters

__position	The number of places to shift.

These should be self-explanatory.

operator[]() [1/2]

```
template<size_t _Nb>
reference std::bitset< _Nb >::operator[] (
            size_t __position)  [inline], [constexpr]
```

Array-indexing support.

Parameters

__position	Index into the bitset.

Returns

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

_GLIBCXX_RESOLVE_LIB_DEFECTS Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme

operator[]() [2/2]

```
template<size_t _Nb>
bool std::bitset< _Nb >::operator[] (
            size_t __position) const  [inline], [constexpr]
```

Array-indexing support.

Parameters

__position	Index into the bitset.

Returns

A bool for a *const bitset*. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

_GLIBCXX_RESOLVE_LIB_DEFECTS Note that this implementation already resolves DR 11 (items 1 and 2), but does not do the range-checking required by that DR's resolution. -pme The DR has since been changed: range-checking is a precondition (users' responsibility), and these functions must not throw. -pme
References _Unchecked_test().

operator^=()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator^= (
            const bitset< _Nb > & __rhs)  [inline], [constexpr], [noexcept]
```

Operations on bitsets.

Parameters

__rhs	A same-sized bitset.

These should be self-explanatory.
References bitset().

operator"|=()

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::operator|= (
            const bitset< _Nb > & __rhs)  [inline], [constexpr], [noexcept]
```

Operations on bitsets.

Parameters

__rhs	A same-sized bitset.

These should be self-explanatory.
References bitset().

operator~()

```
template<size_t _Nb>
bitset< _Nb > std::bitset< _Nb >::operator~ () const  [inline], [constexpr], [noexcept]
```

See the no-argument flip().
References bitset().

reset() [1/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::reset ()  [inline], [constexpr], [noexcept]
```

Sets every bit to false.

reset() [2/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::reset (
            size_t __position)  [inline], [constexpr]
```

Sets a given bit to false.

Parameters

__position	The index of the bit.

Exceptions

std::out_of_range	If *pos* is bigger the size of the set.

Same as writing `set(pos,false)`.
References _Unchecked_reset().

set() [1/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::set ()  [inline], [constexpr], [noexcept]
```

Sets every bit to true.

set() [2/2]

```
template<size_t _Nb>
bitset< _Nb > & std::bitset< _Nb >::set (
            size_t __position,
            bool __val = true)  [inline], [constexpr]
```

Sets a given bit to a particular value.

Parameters

__position	The index of the bit.
__val	Either true or false, defaults to true.

Exceptions

std::out_of_range	If *pos* is bigger the size of the set.

References _Unchecked_set().

size()

```
template<size_t _Nb>
size_t std::bitset< _Nb >::size () const  [inline], [constexpr], [noexcept]
```

Returns the total number of bits.

test()

```
template<size_t _Nb>
bool std::bitset< _Nb >::test (
            size_t __position) const  [inline], [constexpr]
```

Tests the value of a bit.

Parameters

__position	The index of a bit.

Returns

The value at *pos*.

Exceptions

std::out_of_range	If *pos* is bigger the size of the set.

References _Unchecked_test().

to_string()

```
template<size_t _Nb>
template<class _CharT, class _Traits, class _Alloc>
std::basic_string< _CharT, _Traits, _Alloc > std::bitset< _Nb >::to_string () const  [inline],
[constexpr]
```

Returns a character interpretation of the bitset.

Returns

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

to_ulong()

```
template<size_t _Nb>
unsigned long std::bitset< _Nb >::to_ulong () const  [inline], [constexpr]
```

Returns a numerical interpretation of the bitset.

Returns

The integral equivalent of the bits.

Exceptions

std::overflow_error	If there are too many bits to be represented in an `unsigned long`.

The documentation for this class was generated from the following file:

- bitset

6.279 std::tr2::bool_set Class Reference

`#include <bool_set>`

Public Member Functions

- constexpr bool_set ()
- constexpr bool_set (bool __t)
- bool **contains** (bool_set __b) const
- bool equals (bool_set __b) const
- bool is_emptyset () const
- bool is_indeterminate () const
- bool is_singleton () const
- operator bool () const

Static Public Member Functions

- static bool_set **emptyset** ()
- static bool_set **indeterminate** ()

Friends

- bool_set **operator!** (bool_set __b)
- bool_set **operator&** (bool_set __s, bool_set __t)
- template<typename CharT, typename Traits>
  std::basic_ostream< CharT, Traits > & **operator**<< (std::basic_ostream< CharT, Traits > &__out, bool_set __b)
- bool_set **operator==** (bool_set __s, bool_set __t)
- template<typename CharT, typename Traits>
  std::basic_istream< CharT, Traits > & **operator**>> (std::basic_istream< CharT, Traits > &__in, bool_set &__b)
- bool_set **operator^** (bool_set __s, bool_set __t)
- bool_set **operator|** (bool_set __s, bool_set __t)

6.279.1 Detailed Description

bool_set
See N2136, Bool_set: multi-valued logic by Hervé Brönnimann, Guillaume Melquiond, Sylvain Pion.
The implicit conversion to bool is slippery! I may use the new explicit conversion. This has been specialized in the language so that in contexts requiring a bool the conversion happens implicitly. Thus most objections should be eliminated.

6.279.2 Constructor & Destructor Documentation

bool_set() [1/2]

```
std::tr2::bool_set::bool_set ()  [inline], [constexpr]
```

Default constructor.
Referenced by equals().

bool_set() [2/2]

```
std::tr2::bool_set::bool_set (
            bool __t)  [inline], [constexpr]
```

Constructor from bool.

6.279.3 Member Function Documentation

equals()

```
bool std::tr2::bool_set::equals (
            bool_set __b) const  [inline]
```

Return true if states are equal.
References bool_set().

is_emptyset()

```
bool std::tr2::bool_set::is_emptyset () const  [inline]
```

Return true if this is empty.

is_indeterminate()

```
bool std::tr2::bool_set::is_indeterminate () const  [inline]
```

Return true if this is indeterminate.

is_singleton()

```
bool std::tr2::bool_set::is_singleton () const  [inline]
```

Return true if this is false or true (normal boolean).
Referenced by operator bool().

operator bool()

```
std::tr2::bool_set::operator bool () const  [inline]
```

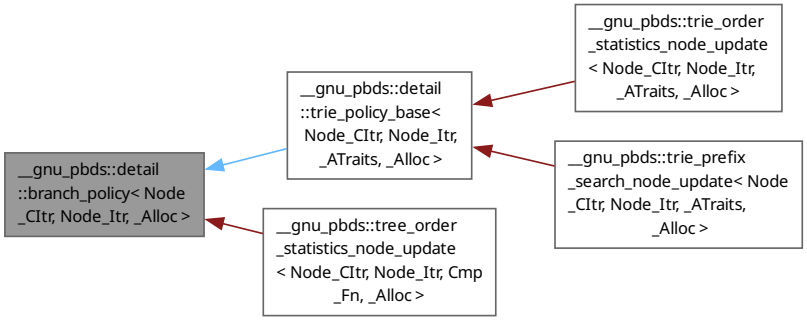
Conversion to bool.
References is_singleton().
The documentation for this class was generated from the following files:

- bool_set
- bool_set.tcc

6.280 __gnu_pbds::detail::branch_policy< Node_Cltr, Node_Itr, _Alloc > Struct Template Reference

```
#include <branch_policy.hpp>
```

Inheritance diagram for __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc >:



Protected Types

- typedef rebind_v::const_pointer **const_pointer**
- typedef rebind_v::const_reference **const_reference**
- typedef Node_Itr::value_type **it_type**
- typedef rebind_k::const_reference **key_const_reference**
- typedef value_type::first_type **key_type**
- typedef remove_const< key_type >::type **rckey_type**
- typedef remove_const< value_type >::type **rcvalue_type**
- typedef rebind_traits< _Alloc, rckey_type > **rebind_k**
- typedef rebind_traits< _Alloc, rcvalue_type > **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits< it_type >::value_type **value_type**

Protected Member Functions

- virtual it_type **end** ()=0
- it_type **end_iterator** () const

Static Protected Member Functions

- static key_const_reference **extract_key** (const_reference r_val)

6.280.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _Alloc>
struct __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc >

Primary template, base class for branch structure policies.
The documentation for this struct was generated from the following file:

- branch_policy.hpp

6.281 __gnu_pbds::detail::branch_policy< Node_CItr, Node_CItr, _Alloc > Struct Template Reference

`#include <branch_policy.hpp>`

Protected Types

- typedef rebind_v::const_pointer **const_pointer**
- typedef rebind_v::const_reference **const_reference**
- typedef Node_CItr::value_type **it_type**
- typedef rebind_v::const_reference **key_const_reference**
- typedef value_type **key_type**
- typedef remove_const< key_type >::type **rckey_type**
- typedef remove_const< value_type >::type **rcvalue_type**
- typedef rebind_traits< _Alloc, rckey_type > **rebind_k**
- typedef rebind_traits< _Alloc, rcvalue_type > **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits< it_type >::value_type **value_type**

Protected Member Functions

- virtual it_type **end** () const =0
- it_type **end_iterator** () const

Static Protected Member Functions

- static key_const_reference **extract_key** (const_reference r_val)

6.281.1 Detailed Description

template<typename Node_CItr, typename _Alloc>
struct __gnu_pbds::detail::branch_policy< Node_CItr, Node_CItr, _Alloc >

Specialization for const iterators.
The documentation for this struct was generated from the following file:

- branch_policy.hpp

6.282 std::cauchy_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **cauchy_distribution** (_RealType __a, _RealType __b=1.0)
- **cauchy_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)

- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType **a** () const
- _RealType **b** () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const cauchy_distribution &__d1, const cauchy_distribution &__d2)

6.282.1 Detailed Description

template<typename _RealType = double>
class std::cauchy_distribution< _RealType >

A cauchy_distribution random number distribution.
The formula for the normal probability mass function is $p(x|a,b) = (\pi b(1+(\frac{x-a}{b})^2))^{-1}$

Since

C++11

6.282.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::cauchy_distribution< _RealType >::result_type
```

The type of the range of the distribution.

6.282.3 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::cauchy_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::cauchy_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::cauchy_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::cauchy_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
template<typename _RealType = double>
void std::cauchy_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::cauchy_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

6.282.4 Friends And Related Symbol Documentation

operator==

```
template<typename _RealType = double>
bool operator== (
            const cauchy_distribution< _RealType > & __d1,
            const cauchy_distribution< _RealType > & __d2)  [friend]
```

Return true if two Cauchy distributions have the same parameters.
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.283 __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference

```
#include <hash_policy.hpp>
```

Public Types

- enum { external_load_access }
- typedef Size_Type **size_type**

Public Member Functions

- cc_hash_max_collision_check_resize_trigger (float load=0.5)
- float get_load () const
- void set_load (float load)
- void **swap** (cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type > &other)

Protected Member Functions

- bool is_grow_needed (size_type size, size_type num_entries) const
- bool is_resize_needed () const
- void notify_cleared ()
- void notify_erase_search_collision ()
- void notify_erase_search_end ()
- void notify_erase_search_start ()
- void notify_erased (size_type num_entries)
- void notify_externally_resized (size_type new_size)
- void notify_find_search_collision ()
- void notify_find_search_end ()
- void notify_find_search_start ()
- void notify_insert_search_collision ()
- void notify_insert_search_end ()
- void notify_insert_search_start ()
- void notify_inserted (size_type num_entries)
- void notify_resized (size_type new_size)

6.283.1 Detailed Description

template<bool External_Load_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >

A resize trigger policy based on collision checks. It keeps the simulated load factor lower than some given load factor.

6.283.2 Member Enumeration Documentation

anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum
```

Enumerator

external_load_access	Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

6.283.3 Constructor & Destructor Documentation

cc_hash_max_collision_check_resize_trigger()

```
template<bool External_Load_Access, typename Size_Type>
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >::cc_↩
hash_max_collision_check_resize_trigger (
            float load = 0.5)
```

Default constructor, or constructor taking load, a __load factor which it will attempt to maintain.

6.283.4 Member Function Documentation

get_load()

```
template<bool External_Load_Access, typename Size_Type>
float __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::get_load () const  [inline]
```

Returns the current load.

is_grow_needed()

```
template<bool External_Load_Access, typename Size_Type>
bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::is_grow_needed (
            size_type size,
            size_type num_entries) const  [inline], [protected]
```

Queries whether a grow is needed. This method is called only if this object indicated is needed.

is_resize_needed()

```
template<bool External_Load_Access, typename Size_Type>
bool __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::is_resize_needed () const  [inline], [protected]
```

Queries whether a resize is needed.

notify_cleared()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_cleared ()  [protected]
```

Notifies the table was cleared.
References notify_resized().

notify_erase_search_collision()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_erase_search_collision ()  [inline], [protected]
```

Notifies a search encountered a collision.

notify_erase_search_end()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_erase_search_end ()  [inline], [protected]
```

Notifies a search ended.

notify_erase_search_start()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_erase_search_start ()  [inline], [protected]
```

Notifies an erase search started.

notify_erased()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_erased (
            size_type num_entries)  [inline], [protected]
```

Notifies an element was erased.

notify_externally_resized()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_externally_resized (
            size_type new_size)  [protected]
```

Notifies the table was resized externally.

notify_find_search_collision()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_find_search_collision ()  [inline], [protected]
```

Notifies a search encountered a collision.

notify_find_search_end()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_find_search_end ()  [inline], [protected]
```

Notifies a search ended.

notify_find_search_start()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_find_search_start ()  [inline], [protected]
```

Notifies a find search started.

notify_insert_search_collision()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_insert_search_collision ()  [inline], [protected]
```

Notifies a search encountered a collision.

notify_insert_search_end()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_insert_search_end ()  [inline], [protected]
```

Notifies a search ended.

notify_insert_search_start()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_insert_search_start ()  [inline], [protected]
```

Notifies an insert search started.

notify_inserted()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_inserted (
            size_type num_entries)  [inline], [protected]
```

Notifies an element was inserted.

notify_resized()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::notify_resized (
            size_type new_size)  [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.
Referenced by notify_cleared().

set_load()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type
>::set_load (
            float load)
```

Sets the load; does not resize the container.
The documentation for this class was generated from the following file:

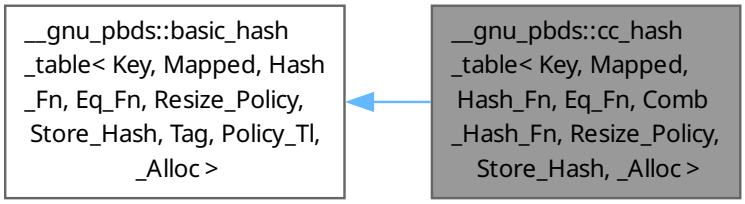
- hash_policy.hpp

6.284 __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc > Class Template Reference

```
#include <assoc_container.hpp>
```

Inheritance diagram for __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy,

Store_Hash, _Alloc >:



Public Types

- typedef Comb_Hash_Fn **comb_hash_fn**
- typedef cc_hash_tag **container_category**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef Resize_Policy **resize_policy**

Public Member Functions

- cc_hash_table ()
- **cc_hash_table** (const cc_hash_table &other)
- cc_hash_table (const hash_fn &h)
- cc_hash_table (const hash_fn &h, const eq_fn &e)
- cc_hash_table (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)
- cc_hash_table (const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)
- template<typename It>
  cc_hash_table (It first, It last)
- template<typename It>
  cc_hash_table (It first, It last, const hash_fn &h)
- template<typename It>
  cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e)
- template<typename It>
  cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch)
- template<typename It>
  cc_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_hash_fn &ch, const resize_policy &rp)
- cc_hash_table & **operator=** (const cc_hash_table &other)
- void **swap** (cc_hash_table &other)

6.284.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>> class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >

A collision-chaining hash-based associative container.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor.
Eq_Fn	Equal functor.
Comb_Hash_Fn	Combining hash functor. If Hash_Fn is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.)
Resize_Policy	Resizes hash.
Store_Hash	Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true
_Alloc	Allocator type.

Base tag choices are: cc_hash_tag.
Base is basic_hash_table.

6.284.2 Constructor & Destructor Documentation

cc_hash_table() [1/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash↩
_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_↩
resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc
= std::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table ()  [inline]
```

Default constructor.

cc_hash_table() [2/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash↩
_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_↩
resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc
= std::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            const hash_fn & h)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the Hash_Fn object of the container object.

cc_hash_table() [3/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash↩
_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_↩
resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc
= std::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            const hash_fn & h,
            const eq_fn & e)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, and r_eq_fn will be copied by the eq_fn object of the container object.

cc_hash_table() [4/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash↩
_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_↩
resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc
= std::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            const hash_fn & h,
            const eq_fn & e,
            const comb_hash_fn & ch)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, and r_comb_hash_fn will be copied by the comb_hash_fn object of the container object.

cc_hash_table() [5/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash↩
_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_↩
resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc
= std::allocator<char>>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            const hash_fn & h,
            const eq_fn & e,
            const comb_hash_fn & ch,
            const resize_policy & rp)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_hash_fn will be copied by the comb_hash_fn object of the container object, and r_resize_policy will be copied by the resize_policy object of the container object.

cc_hash_table() [6/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash↩
_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_↩
resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc
= std::allocator<char>>
template<typename It>
```

```
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            It first,
            It last)  [inline]
```

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

cc_hash_table() [7/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash↩
_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_↩
resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc
= std::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            It first,
            It last,
            const hash_fn & h)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object.

cc_hash_table() [8/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash↩
_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_↩
resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc
= std::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            It first,
            It last,
            const hash_fn & h,
            const eq_fn & e)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, and r_eq_fn will be copied by the eq_fn object of the container object.

cc_hash_table() [9/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash↩
_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_↩
resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc
= std::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash,
_Alloc >::cc_hash_table (
            It first,
            It last,
            const hash_fn & h,
```

```
            const eq_fn & e,
            const comb_hash_fn & ch)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, and r_comb_hash_fn will be copied by the comb_hash_fn object of the container object.

cc_hash_table() [10/10]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Hash_Fn = detail::default_comb_hash_fn::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Hash_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >::cc_hash_table (
            It first,
            It last,
            const hash_fn & h,
            const eq_fn & e,
            const comb_hash_fn & ch,
            const resize_policy & rp)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_hash_fn will be copied by the comb_hash_fn object of the container object, and r_resize_policy will be copied by the resize_policy object of the container object.

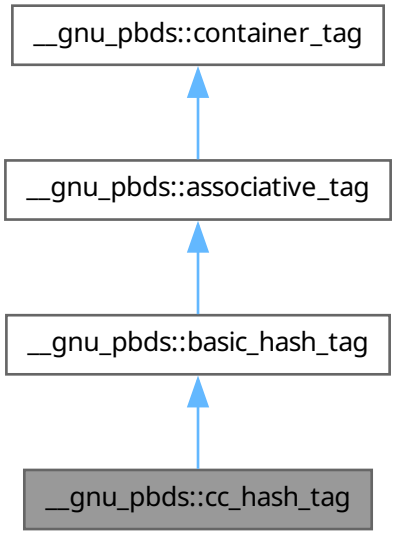
The documentation for this class was generated from the following file:

- assoc_container.hpp

6.285 __gnu_pbds::cc_hash_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::cc_hash_tag:



6.285.1 Detailed Description

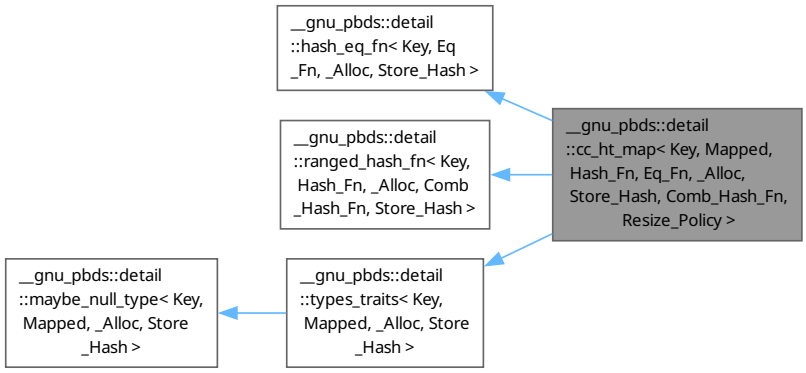
Collision-chaining hash.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.286 __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > Class Template Reference

`#include <cc_ht_map_.hpp>`
Inheritance diagram for __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb↩

_Hash_Fn, Resize_Policy >:



Public Types

- enum { **store_hash** }
- typedef _Alloc **allocator_type**
- typedef Comb_Hash_Fn **comb_hash_fn**
- typedef const_iterator_ **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef iterator_ **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef point_const_iterator_ **point_const_iterator**
- typedef point_iterator_ **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef Resize_Policy **resize_policy**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **cc_ht_map** (const cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > &)
- **cc_ht_map** (const Hash_Fn &)
- **cc_ht_map** (const Hash_Fn &, const Eq_Fn &)
- **cc_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Hash_Fn &)
- **cc_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Hash_Fn &, const Resize_Policy &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It>
  void **copy_from_range** (It, It)
- bool empty () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- point_iterator **find_end** ()
- point_const_iterator **find_end** () const
- Comb_Hash_Fn & get_comb_hash_fn ()
- const Comb_Hash_Fn & get_comb_hash_fn () const
- Eq_Fn & get_eq_fn ()
- const Eq_Fn & get_eq_fn () const
- Hash_Fn & get_hash_fn ()
- const Hash_Fn & get_hash_fn () const
- Resize_Policy & get_resize_policy ()
- const Resize_Policy & get_resize_policy () const
- void **initialize** ()
- std::pair< point_iterator, bool > **insert** (const_reference r_val)
- size_type **max_size** () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- size_type **size** () const
- void **swap** (cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy > &)

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Friends

- class **const_iterator_**
- class **iterator_**

6.286.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >

A collision-chaining hash-based container.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor. Default is __gnu_cxx::hash.
Eq_Fn	Equal functor. Default std::equal_to<Key>
_Alloc	Allocator type.
Store_Hash	If key type stores extra metadata. Defaults to false.
Comb_Hash_Fn	Combining hash functor. If Hash_Fn is not null_type, then this is the ranged-hash functor; otherwise, this is the range-hashing functor. XXX(See Design::Hash-Based Containers::Hash Policies.) Default direct_mask_range_hashing.
Resize_Policy	Resizes hash. Defaults to hash_standard_resize_policy, using hash_exponential_size_policy and hash_load_check_resize_trigger.

Bases are: detail::hash_eq_fn, Resize_Policy, detail::ranged_hash_fn, detail::types_traits. (Optional: detail::debug_↩ map_base.)

6.286.2 Member Enumeration Documentation

anonymous enum

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
anonymous enum
```

Value stores hash, true or false.

6.286.3 Member Function Documentation

empty()

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
bool __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_↩
Fn, Resize_Policy >::empty () const  [inline], [nodiscard]
```

True if size() == 0.

get_comb_hash_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn ()
```

Return current comb_hash_fn.

get_comb_hash_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Comb_Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↩
Hash, Comb_Hash_Fn, Resize_Policy >::get_comb_hash_fn () const
```

Return current const comb_hash_fn.

get_eq_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩
Hash_Fn, Resize_Policy >::get_eq_fn ()
```

Return current eq_fn.

get_eq_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Eq_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_eq_fn () const
```

Return current const eq_fn.

get_hash_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩
Hash_Fn, Resize_Policy >::get_hash_fn ()
```

Return current hash_fn.

get_hash_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Hash_Fn & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_hash_fn () const
```

Return current const hash_fn.

get_resize_policy() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Hash_Fn, Resize_Policy >::get_resize_policy ()
```

Return current resize_policy.

get_resize_policy() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Hash_Fn, typename Resize_Policy>
const Resize_Policy & __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↩
Hash, Comb_Hash_Fn, Resize_Policy >::get_resize_policy () const
```

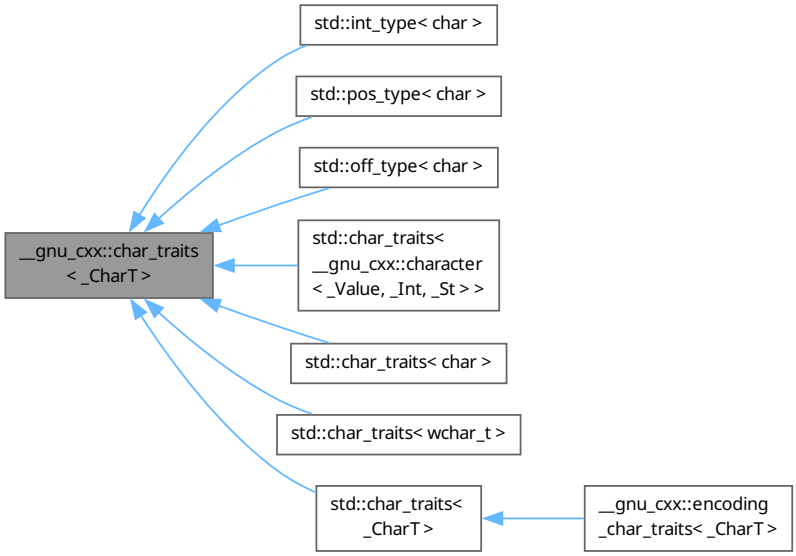
Return current const resize_policy.

The documentation for this class was generated from the following file:

- cc_ht_map_.hpp

6.287 __gnu_cxx::char_traits< _CharT > Struct Template Reference

`#include <char_traits.h>`
Inheritance diagram for __gnu_cxx::char_traits< _CharT >:



Public Types

- typedef _CharT **char_type**
- using **comparison_category**
- typedef _Char_types< _CharT >::int_type **int_type**
- typedef _Char_types< _CharT >::off_type **off_type**
- typedef _Char_types< _CharT >::pos_type **pos_type**
- typedef _Char_types< _CharT >::state_type **state_type**

Static Public Member Functions

- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)

- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)

6.287.1 Detailed Description

template<typename _CharT>
struct __gnu_cxx::char_traits< _CharT >

Base class used to implement std::char_traits.

Note

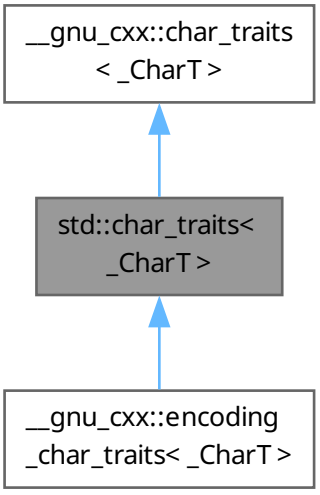
For any given actual character type, this definition is probably wrong. (Most of the member functions are likely to be right, but the int_type and state_type typedefs, and the eof() member function, are likely to be wrong.) The reason this class exists is so users can specialize it. Classes in namespace std may not be specialized for fundamental types, but classes in namespace __gnu_cxx may be.

See https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character_types for advice on how to make use of this class for *unusual* character types. Also, check out include/ext/pod_char_traits.h.
The documentation for this struct was generated from the following file:

- char_traits.h

6.288 std::char_traits< _CharT > Struct Template Reference

`#include <char_traits.h>`
Inheritance diagram for std::char_traits< _CharT >:



Public Types

- typedef _CharT **char_type**

- using **comparison_category**
- typedef _Char_types< _CharT >::int_type **int_type**
- typedef _Char_types< _CharT >::off_type **off_type**
- typedef _Char_types< _CharT >::pos_type **pos_type**
- typedef _Char_types< _CharT >::state_type **state_type**

Static Public Member Functions

- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)

6.288.1 Detailed Description

template<**typename _CharT**>
struct std::char_traits< **_CharT** >

Basis for explicit traits specializations.

Note

For any given actual character type, this definition is probably wrong. Since this is just a thin wrapper around __gnu_cxx::char_traits, it is possible to achieve a more appropriate definition by specializing __gnu_cxx::char_traits.

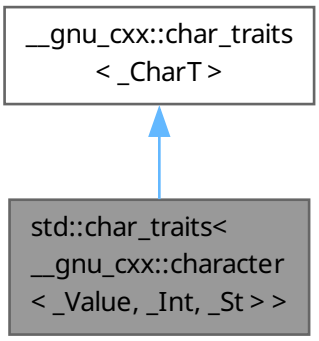
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/strings.html#strings.string.character_types for advice on how to make use of this class for *unusual* character types. Also, check out include/ext/pod_char_traits.h.
The documentation for this struct was generated from the following file:

- char_traits.h

6.289 std::char_traits< __gnu_cxx::character< _Value, _Int, _St > > Struct Template Reference

`#include <pod_char_traits.h>`

Inheritance diagram for std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >:



Public Types

- typedef __gnu_cxx::character< _Value, _Int, _St > **char_type**
- typedef __gnu_cxx::character< _Value, _Int, _St > **char_type**
- using **comparison_category**
- using **comparison_category**
- typedef _Char_types< __gnu_cxx::character< _Value, _Int, _St > >::int_type **int_type**
- typedef char_type::int_type **int_type**
- typedef _Char_types< __gnu_cxx::character< _Value, _Int, _St > >::off_type **off_type**
- typedef streamoff **off_type**
- typedef _Char_types< __gnu_cxx::character< _Value, _Int, _St > >::pos_type **pos_type**
- typedef fpos< state_type > **pos_type**
- typedef _Char_types< __gnu_cxx::character< _Value, _Int, _St > >::state_type **state_type**
- typedef char_type::state_type **state_type**

Static Public Member Functions

- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static void **assign** (char_type &__c1, const char_type &__c2)
- static char_type * **assign** (char_type *__s, size_t __n, char_type __a)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static int **compare** (const char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static char_type * **copy** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)

- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static const char_type * **find** (const char_type *__s, size_t __n, const char_type &__a)
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr std::size_t **length** (const char_type *__s)
- static size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static char_type * **move** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr int_type **not_eof** (const int_type &__c)
- static int_type **not_eof** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static char_type **to_char_type** (const int_type &__i)
- static constexpr int_type **to_int_type** (const char_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)
- static int_type **to_int_type** (const char_type &__c)

6.289.1 Detailed Description

template<**typename _Value, typename _Int, typename _St**>
struct std::char_traits< **__gnu_cxx::character**< **_Value, _Int, _St** > >

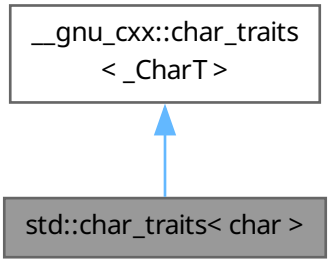
char_traits<__gnu_cxx::character> specialization.
The documentation for this struct was generated from the following file:

- pod_char_traits.h

6.290 std::char_traits< char > Struct Reference

`#include <char_traits.h>`

Inheritance diagram for std::char_traits< char >:



Public Types

- typedef char **char_type**
- typedef char **char_type**
- using **comparison_category**
- using **comparison_category**
- typedef int **int_type**
- typedef _Char_types< char >::int_type **int_type**
- typedef streamoff **off_type**
- typedef _Char_types< char >::off_type **off_type**
- typedef streampos **pos_type**
- typedef _Char_types< char >::pos_type **pos_type**
- typedef mbstate_t **state_type**
- typedef _Char_types< char >::state_type **state_type**

Static Public Member Functions

- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr void **assign** (char_type &__c1, const char_type &__c2) noexcept
- static constexpr char_type * **assign** (char_type *__s, size_t __n, char_type __a)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr int_type **eof** () noexcept
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2) noexcept

- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2) noexcept
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr const char_type * **find** (const char_type *__s, size_t __n, const char_type &__a)
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr size_t **length** (const char_type *__s)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2) noexcept
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr int_type **not_eof** (const int_type &__c) noexcept
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c) noexcept
- static constexpr int_type **to_int_type** (const char_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c) noexcept

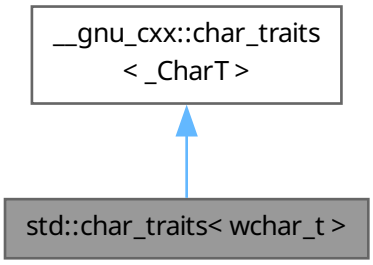
6.290.1 Detailed Description

21.1.3.1 char_traits specializations
The documentation for this struct was generated from the following file:

- char_traits.h

6.291 std::char_traits< wchar_t > Struct Reference

`#include <char_traits.h>`
Inheritance diagram for std::char_traits< wchar_t >:



Public Types

- typedef wchar_t **char_type**
- typedef wchar_t **char_type**
- using **comparison_category**
- using **comparison_category**
- typedef wint_t **int_type**
- typedef _Char_types< wchar_t >::int_type **int_type**
- typedef streamoff **off_type**
- typedef _Char_types< wchar_t >::off_type **off_type**
- typedef wstreampos **pos_type**
- typedef _Char_types< wchar_t >::pos_type **pos_type**
- typedef mbstate_t **state_type**
- typedef _Char_types< wchar_t >::state_type **state_type**

Static Public Member Functions

- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr void **assign** (char_type &__c1, const char_type &__c2) noexcept
- static constexpr char_type * **assign** (char_type *__s, size_t __n, char_type __a)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr int_type **eof** () noexcept
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2) noexcept
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2) noexcept
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr const char_type * **find** (const char_type *__s, size_t __n, const char_type &__a)
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr size_t **length** (const char_type *__s)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2) noexcept
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, size_t __n)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr int_type **not_eof** (const int_type &__c) noexcept

- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c) noexcept
- static constexpr int_type **to_int_type** (const char_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c) noexcept

6.291.1 Detailed Description

21.1.3.2 char_traits specializations
The documentation for this struct was generated from the following file:

- char_traits.h

6.292 __gnu_cxx::character< _Value, _Int, _St > Struct Template Reference

`#include <pod_char_traits.h>`

Public Types

- typedef character< _Value, _Int, _St > **char_type**
- typedef _Int **int_type**
- typedef _St **state_type**
- typedef _Value **value_type**

Static Public Member Functions

- template<typename V2>
  static char_type **from** (const V2 &v)
- template<typename V2>
  static V2 **to** (const char_type &c)

Public Attributes

- value_type **value**

6.292.1 Detailed Description

template<typename _Value, typename _Int, typename _St = std::mbstate_t>
struct __gnu_cxx::character< _Value, _Int, _St >

A POD class that serves as a character abstraction class.
The documentation for this struct was generated from the following file:

- pod_char_traits.h

6.293 std::chi_squared_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **chi_squared_distribution** (_RealType __n)
- **chi_squared_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- _RealType **n** () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::chi_squared_distribution< _RealType1 > &__x)
- bool operator== (const chi_squared_distribution &__d1, const chi_squared_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::chi_squared_distribution< _RealType1 > &__x)

6.293.1 Detailed Description

template<typename _RealType = double>
class std::chi_squared_distribution< _RealType >

A chi_squared_distribution random number distribution.
The formula for the normal probability mass function is $p(x|n) = \frac{x^{(n/2)-1}e^{-x/2}}{\Gamma(n/2)2^{n/2}}$

Since

C++11

6.293.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::chi_squared_distribution< _RealType >::result_type
```

The type of the range of the distribution.

6.293.3 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::chi_squared_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::chi_squared_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::chi_squared_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.

param() [1/2]

```
template<typename _RealType = double>
param_type std::chi_squared_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::chi_squared_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::chi_squared_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

6.293.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::chi_squared_distribution< _RealType1 > & __x)  [friend]
```

Inserts a chi_squared_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A chi_squared_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
            const chi_squared_distribution< _RealType > & __d1,
            const chi_squared_distribution< _RealType > & __d2)  [friend]
```

Return true if two Chi-squared distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::chi_squared_distribution< _RealType1 > & __x)  [friend]
```

Extracts a chi_squared_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A chi_squared_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

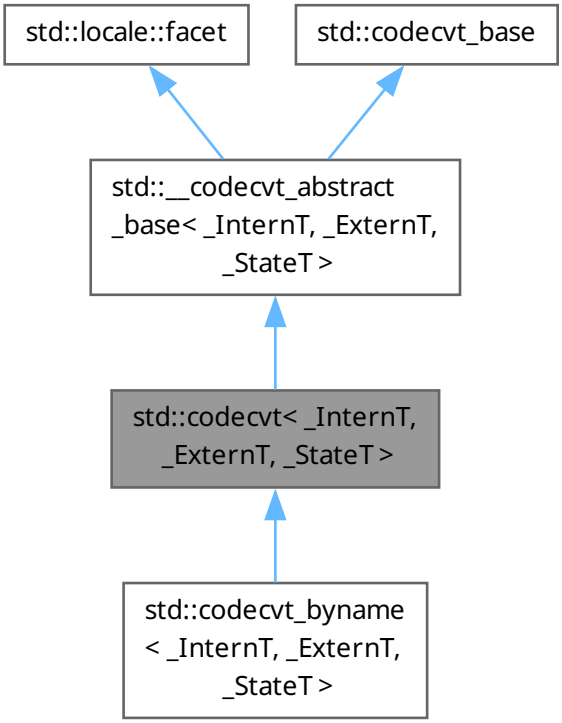
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.294 std::codecvt< _InternT, _ExternT, _StateT > Class Template Reference

`#include <codecvt.h>`

Inheritance diagram for std::codecvt< _InternT, _ExternT, _StateT >:



Public Types

- typedef _ExternT **extern_type**
- typedef _InternT **intern_type**
- typedef codecvt_base::result **result**
- typedef _StateT **state_type**

Public Member Functions

- **codecvt** (__c_locale __cloc, size_t __refs=0)
- **codecvt** (size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()

- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

6.294.1 Detailed Description

template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt< _InternT, _ExternT, _StateT >

Primary class template codecvt.
NB: Generic, mostly useless implementation.

6.294.2 Member Function Documentation

do_always_noconv()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual bool std::codecvt< _InternT, _ExternT, _StateT >::do_always_noconv () const throw ( )
[protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_encoding()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_length()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_max_length()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
```

```
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

References do_out().

unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt↩ _base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

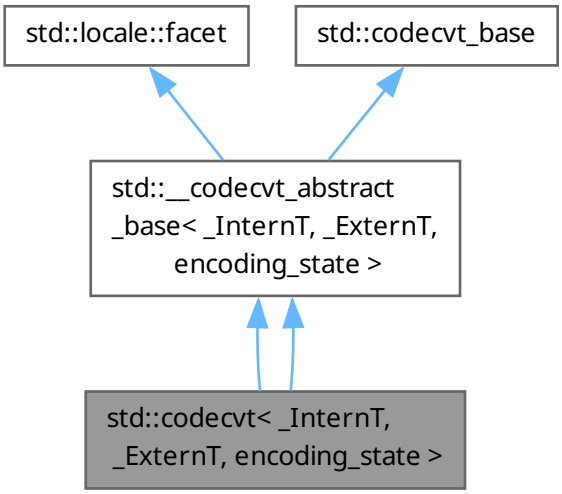
codecvt_base::result.

The documentation for this class was generated from the following file:

- codecvt.h

6.295 std::codecvt< _InternT, _ExternT, encoding_state > Class Template Reference

`#include <codecvt_specializations.h>`
Inheritance diagram for std::codecvt< _InternT, _ExternT, encoding_state >:



Public Types

- typedef state_type::descriptor_type **descriptor_type**

- typedef _ExternT **extern_type**
- typedef _InternT **intern_type**
- typedef codecvt_base::result **result**
- typedef __gnu_cxx::encoding_state **state_type**

Public Member Functions

- **codecvt** (size_t __refs=0)
- **codecvt** (state_type &__enc, size_t __refs=0)
- bool **always_noconv** () const throw ()
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to↩ _next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)

- static __c_locale **_S_get_c_locale** ()
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

6.295.1 Detailed Description

template<typename _InternT, typename _ExternT>
class std::codecvt< _InternT, _ExternT, encoding_state >

codecvt<InternT, _ExternT, encoding_state> specialization.

6.295.2 Member Function Documentation

do_always_noconv()

```
template<typename _InternT, typename _ExternT>
bool std::codecvt< _InternT, _ExternT, encoding_state >::do_always_noconv () const throw ( )
[protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_encoding()

```
template<typename _InternT, typename _ExternT>
int std::codecvt< _InternT, _ExternT, encoding_state >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_in()

```
template<typename _InternT, typename _ExternT>
codecvt_base::result std::codecvt< _InternT, _ExternT, encoding_state >::do_in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_length()

```
template<typename _InternT, typename _ExternT>
int std::codecvt< _InternT, _ExternT, encoding_state >::do_length (
             state_type & ,
             const extern_type * __from,
             const extern_type * __end,
             size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_max_length()

```
template<typename _InternT, typename _ExternT>
int std::codecvt< _InternT, _ExternT, encoding_state >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_out()

```
template<typename _InternT, typename _ExternT>
codecvt_base::result std::codecvt< _InternT, _ExternT, encoding_state >::do_out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

do_unshift()

```
template<typename _InternT, typename _ExternT>
codecvt_base::result std::codecvt< _InternT, _ExternT, encoding_state >::do_unshift (
             state_type & __state,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >.

in() [1/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

in() [2/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [inline]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.

__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [1/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [2/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [1/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [2/2]

```
result std::__codecvt_abstract_base< _InternT, _ExternT, encoding_state >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Reset conversion state.
Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().
For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.
The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

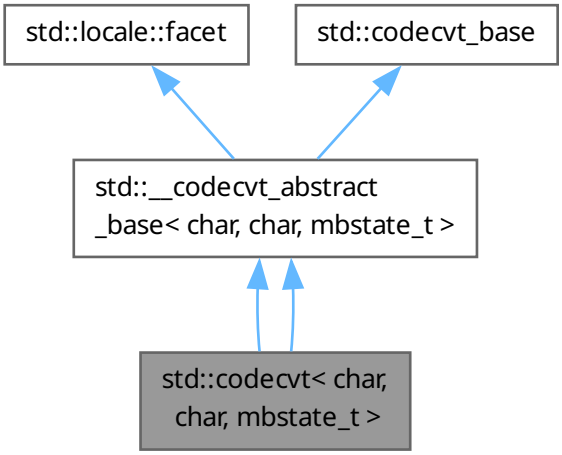
The documentation for this class was generated from the following file:

- codecvt_specializations.h

6.296 std::codecvt< char, char, mbstate_t > Class Reference

`#include <codecvt.h>`

Inheritance diagram for std::codecvt< char, char, mbstate_t >:



Public Types

- typedef char **extern_type**
- typedef char **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**

Public Member Functions

- **codecvt** (__c_locale __cloc, size_t __refs=0)
- **codecvt** (size_t __refs=0)
- bool **always_noconv** () const throw ()
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

Friends

- class **messages**< **char** >

6.296.1 Detailed Description

class codecvt<char, char, mbstate_t> specialization.

6.296.2 Member Function Documentation

do_always_noconv()

```
virtual bool std::codecvt< char, char, mbstate_t >::do_always_noconv () const throw ( )   [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_encoding()

```
virtual int std::codecvt< char, char, mbstate_t >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_in()

```
virtual result std::codecvt< char, char, mbstate_t >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_length()

```
virtual int std::codecvt< char, char, mbstate_t >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_max_length()

```
virtual int std::codecvt< char, char, mbstate_t >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_out()

```
virtual result std::codecvt< char, char, mbstate_t >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

do_unshift()

```
virtual result std::codecvt< char, char, mbstate_t >::do_unshift (
            state_type & __state,
```

```
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char, char, mbstate_t >.

in() [1/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

in() [2/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
```

```
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [1/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the

output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [2/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.

__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [1/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [2/2]

```
result std::__codecvt_abstract_base< char, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

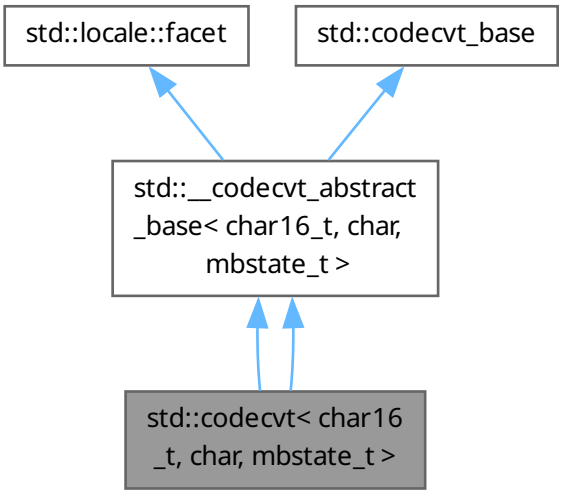
codecvt_base::result.

The documentation for this class was generated from the following file:

- codecvt.h

6.297 std::codecvt< char16_t, char, mbstate_t > Class Reference

`#include <codecvt.h>`
Inheritance diagram for std::codecvt< char16_t, char, mbstate_t >:



Public Types

- typedef char **extern_type**

- typedef char16_t **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**

Public Member Functions

- **codecvt** (size_t __refs=0)
- bool **always_noconv** () const throw ()
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to↩ _next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static __c_locale **_S_get_c_locale** ()

- static const char * **_S_get_c_name** () throw ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

6.297.1 Detailed Description

Class codecvt<char16_t, char, mbstate_t> specialization.
Converts between UTF-16 and UTF-8.

6.297.2 Member Function Documentation

do_always_noconv()

```
virtual bool std::codecvt< char16_t, char, mbstate_t >::do_always_noconv () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_encoding()

```
virtual int std::codecvt< char16_t, char, mbstate_t >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_in()

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_length()

```
virtual int std::codecvt< char16_t, char, mbstate_t >::do_length (
             state_type & ,
             const extern_type * __from,
             const extern_type * __end,
             size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_max_length()

```
virtual int std::codecvt< char16_t, char, mbstate_t >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_out()

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

do_unshift()

```
virtual result std::codecvt< char16_t, char, mbstate_t >::do_unshift (
             state_type & __state,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char16_t, char, mbstate_t >.

in() [1/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.
Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.
The source and destination character sets are determined by the facet's locale, internal and external types.
The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.
The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.
The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.

__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

in() [2/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [inline]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [1/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [2/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
```

```
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [1/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [2/2]

```
result std::__codecvt_abstract_base< char16_t, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Reset conversion state.
Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().
For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.
The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

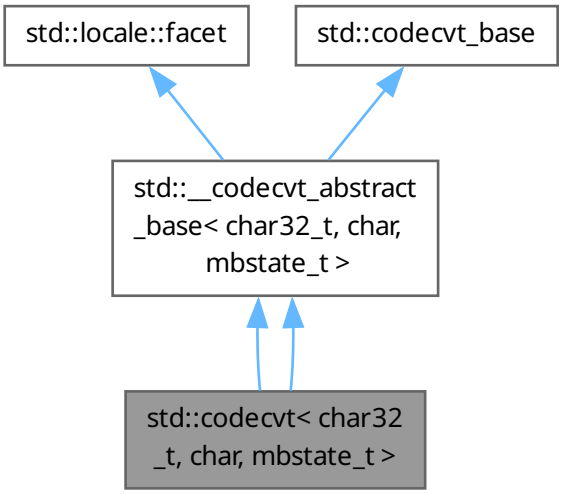
The documentation for this class was generated from the following file:

- codecvt.h

6.298 std::codecvt< char32_t, char, mbstate_t > Class Reference

```
#include <codecvt.h>
```

Inheritance diagram for std::codecvt< char32_t, char, mbstate_t >:



Public Types

- typedef char **extern_type**
- typedef char32_t **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**

Public Member Functions

- **codecvt** (size_t __refs=0)
- bool **always_noconv** () const throw ()
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to↩ _next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

6.298.1 Detailed Description

Class codecvt<char32_t, char, mbstate_t> specialization.
Converts between UTF-32 and UTF-8.

6.298.2 Member Function Documentation

do_always_noconv()

```
virtual bool std::codecvt< char32_t, char, mbstate_t >::do_always_noconv () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_encoding()

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_in()

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_length()

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_max_length()

```
virtual int std::codecvt< char32_t, char, mbstate_t >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_out()

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

do_unshift()

```
virtual result std::codecvt< char32_t, char, mbstate_t >::do_unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< char32_t, char, mbstate_t >.

in() [1/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

in() [2/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.
The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.
The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.
The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [1/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.
The source and destination character sets are determined by the facet's locale, internal and external types.
The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.
The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.
The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [2/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [1/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [2/2]

```
result std::__codecvt_abstract_base< char32_t, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

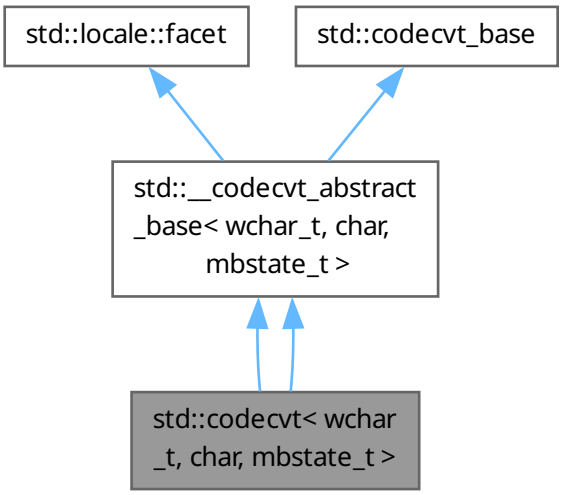
codecvt_base::result.

The documentation for this class was generated from the following file:

- codecvt.h

6.299 std::codecvt< wchar_t, char, mbstate_t > Class Reference

`#include <codecvt.h>`
Inheritance diagram for std::codecvt< wchar_t, char, mbstate_t >:



Public Types

- typedef char **extern_type**
- typedef wchar_t **intern_type**
- typedef codecvt_base::result **result**
- typedef mbstate_t **state_type**

Public Member Functions

- **codecvt** (__c_locale __cloc, size_t __refs=0)
- **codecvt** (size_t __refs=0)
- bool **always_noconv** () const throw ()
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

Friends

- class **messages**< **wchar_t** >

6.299.1 Detailed Description

Class codecvt<wchar_t, char, mbstate_t> specialization.
Converts between narrow and wide characters in the native character set

6.299.2 Member Function Documentation

do_always_noconv()

```
virtual bool std::codecvt< wchar_t, char, mbstate_t >::do_always_noconv () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_encoding()

```
virtual int std::codecvt< wchar_t, char, mbstate_t >::do_encoding () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_in()

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_length()

```
virtual int std::codecvt< wchar_t, char, mbstate_t >::do_length (
             state_type & ,
             const extern_type * __from,
             const extern_type * __end,
             size_t __max) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_max_length()

```
virtual int std::codecvt< wchar_t, char, mbstate_t >::do_max_length () const throw ( )   [protected],
[virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_out()

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [protected], [virtual]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

do_unshift()

```
virtual result std::codecvt< wchar_t, char, mbstate_t >::do_unshift (
             state_type & __state,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [protected], [virtual]
```

Implements std::__codecvt_abstract_base< wchar_t, char, mbstate_t >.

in() [1/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.
Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.
The source and destination character sets are determined by the facet's locale, internal and external types.
The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.
The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.
The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.

__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

in() [2/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::in (
             state_type & __state,
             const extern_type * __from,
             const extern_type * __from_end,
             const extern_type *& __from_next,
             intern_type * __to,
             intern_type * __to_end,
             intern_type *& __to_next) const  [inline]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [1/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
             extern_type * __to_end,
             extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out() [2/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::out (
             state_type & __state,
             const intern_type * __from,
             const intern_type * __from_end,
             const intern_type *& __from_next,
             extern_type * __to,
```

```
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [1/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

unshift() [2/2]

```
result std::__codecvt_abstract_base< wchar_t, char, mbstate_t >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline]
```

Reset conversion state.
Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().
For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.
The source and destination character sets are determined by the facet's locale, internal and external types.
The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

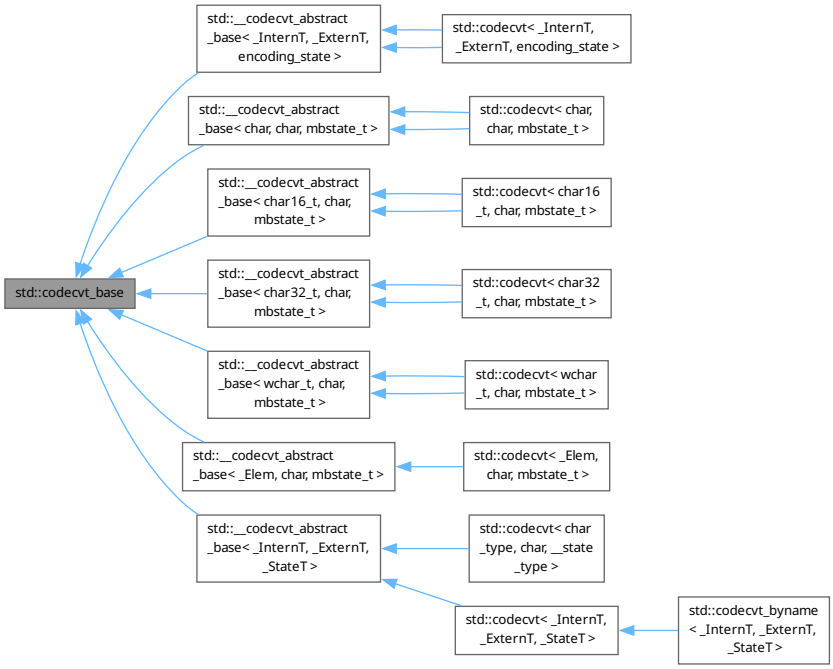
The documentation for this class was generated from the following file:

- codecvt.h

6.300 std::codecvt_base Class Reference

```
#include <codecvt.h>
```

Inheritance diagram for std::codecvt_base:



Public Types

- enum **result** { **ok** , **partial** , **error** , **noconv** }

6.300.1 Detailed Description

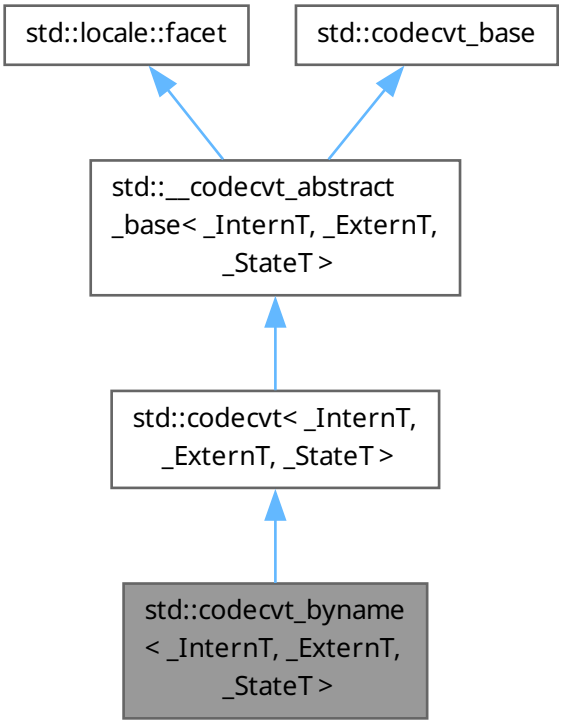
Empty base class for codecvt facet [22.2.1.5].
The documentation for this class was generated from the following file:

- codecvt.h

6.301 std::codecvt_byname< _InternT, _ExternT, _StateT > Class Template Reference

`#include <codecvt.h>`

Inheritance diagram for std::codecvt_byname< _InternT, _ExternT, _StateT >:



Public Types

- typedef _ExternT **extern_type**
- typedef _InternT **intern_type**
- typedef codecvt_base::result **result**
- typedef _StateT **state_type**

Public Member Functions

- **codecvt_byname** (const char *__s, size_t __refs=0)
- **codecvt_byname** (const string &__s, size_t __refs=0)
- bool **always_noconv** () const throw ()
- int **encoding** () const throw ()
- result in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- int **length** (state_type &__state, const extern_type *__from, const extern_type *__end, size_t __max) const
- int **max_length** () const throw ()
- result out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- result unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const

Static Public Attributes

- static locale::id **id**

Protected Member Functions

- virtual bool do_always_noconv () const throw ()
- virtual int do_encoding () const throw ()
- virtual result do_in (state_type &__state, const extern_type *__from, const extern_type *__from_end, const extern_type *&__from_next, intern_type *__to, intern_type *__to_end, intern_type *&__to_next) const
- virtual int do_length (state_type &, const extern_type *__from, const extern_type *__end, size_t __max) const
- virtual int do_max_length () const throw ()
- virtual result do_out (state_type &__state, const intern_type *__from, const intern_type *__from_end, const intern_type *&__from_next, extern_type *__to, extern_type *__to_end, extern_type *&__to_next) const
- virtual result do_unshift (state_type &__state, extern_type *__to, extern_type *__to_end, extern_type *&__to↩ _next) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_codecvt**

6.301.1 Detailed Description

template<typename _InternT, typename _ExternT, typename _StateT>
class std::codecvt_byname< _InternT, _ExternT, _StateT >

class codecvt_byname [22.2.1.6].

6.301.2 Member Function Documentation

do_always_noconv()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual bool std::codecvt< _InternT, _ExternT, _StateT >::do_always_noconv () const throw ( )
[protected], [virtual], [inherited]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_encoding()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_encoding () const throw ( )   [protected],
[virtual], [inherited]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [protected], [virtual], [inherited]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_length()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_length (
            state_type & ,
            const extern_type * __from,
            const extern_type * __end,
            size_t __max) const  [protected], [virtual], [inherited]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_max_length()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual int std::codecvt< _InternT, _ExternT, _StateT >::do_max_length () const throw ( )   [protected],
[virtual], [inherited]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual], [inherited]
```

Convert from internal to external character set.
Converts input string of intern_type to output string of extern_type. This function is a hook for derived classes to change the value returned.

See also

out for more information.

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

do_unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
virtual result std::codecvt< _InternT, _ExternT, _StateT >::do_unshift (
            state_type & __state,
            extern_type * __to,
```

```
            extern_type * __to_end,
            extern_type *& __to_next) const  [protected], [virtual], [inherited]
```

Implements std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >.

in()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::in (
            state_type & __state,
            const extern_type * __from,
            const extern_type * __from_end,
            const extern_type *& __from_next,
            intern_type * __to,
            intern_type * __to_end,
            intern_type *& __to_next) const  [inline], [inherited]
```

Convert from external to internal character set.

Converts input string of extern_type to output string of intern_type. This is analogous to mbsrtowcs. It does this by calling codecvt::do_in.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

out()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::out (
            state_type & __state,
            const intern_type * __from,
            const intern_type * __from_end,
            const intern_type *& __from_next,
```

```
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Convert from internal to external character set.

Converts input string of intern_type to output string of extern_type. This is analogous to wcsrtombs. It does this by calling codecvt::do_out.

The source and destination character sets are determined by the facet's locale, internal and external types.

The characters in [from,from_end) are converted and written to [to,to_end). from_next and to_next are set to point to the character following the last successfully converted character, respectively. If the result needed no conversion, from_next and to_next are not affected.

The *state* argument should be initialized if the input is at the beginning and carried from a previous call if continuing conversion. There are no guarantees about how *state* is used.

The result returned is a member of codecvt_base::result. If all the input is converted, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the input ends early or there is insufficient space in the output, returns codecvt_base::partial. Otherwise the conversion failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__from	Start of input.
__from_end	End of input.
__from_next	Returns start of unconverted data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

codecvt_base::result.

References do_out().

unshift()

```
template<typename _InternT, typename _ExternT, typename _StateT>
result std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >::unshift (
            state_type & __state,
            extern_type * __to,
            extern_type * __to_end,
            extern_type *& __to_next) const  [inline], [inherited]
```

Reset conversion state.

Writes characters to output that would restore *state* to initial conditions. The idea is that if a partial conversion occurs, then the converting the characters written by this function would leave the state in initial conditions, rather than partial conversion state. It does this by calling codecvt::do_unshift().

For example, if 4 external characters always converted to 1 internal character, and input to in() had 6 external characters with state saved, this function would write two characters to the output and set the state to initialized conditions.

The source and destination character sets are determined by the facet's locale, internal and external types.

The result returned is a member of codecvt_base::result. If the state could be reset and data written, returns codecvt_base::ok. If no conversion is necessary, returns codecvt_base::noconv. If the output has insufficient space, returns codecvt_base::partial. Otherwise the reset failed and codecvt_base::error is returned.

Parameters

__state	Persistent conversion state data.
__to	Start of output buffer.
__to_end	End of output buffer.
__to_next	Returns start of unused output area.

Returns

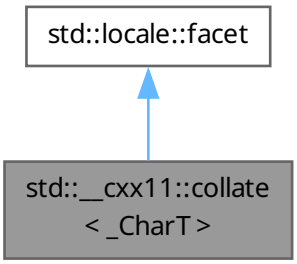
codecvt_base::result.

The documentation for this class was generated from the following file:

- codecvt.h

6.302 std::__cxx11::collate< _CharT > Class Template Reference

`#include <locale_classes.h>`
Inheritance diagram for std::__cxx11::collate< _CharT >:


std::locale::facet
std::__cxx11::collate
< _CharT >
std::__cxx11::collate
< _CharT >
std::__cxx11::collate
< _CharT >

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~collate ()
- virtual int do_compare (const _CharT *__lo1, const _CharT *__hi1, const _CharT *__lo2, const _CharT *__hi2) const
- virtual long do_hash (const _CharT *__lo, const _CharT *__hi) const
- virtual string_type do_transform (const _CharT *__lo, const _CharT *__hi) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_collate**

6.302.1 Detailed Description

template<typename _CharT>
class std::__cxx11::collate< _CharT >

Facet for localized string comparison.
This facet encapsulates the code to compare strings in a localized manner.
The collate template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the collate facet.

6.302.2 Member Typedef Documentation

char_type

```
template<typename _CharT>
typedef _CharT std::__cxx11::collate< _CharT >::char_type
```

Public typedefs.

string_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::__cxx11::collate< _CharT >::string_type
```

Public typedefs.

6.302.3 Constructor & Destructor Documentation

collate() [1/2]

```
template<typename _CharT>
std::__cxx11::collate< _CharT >::collate (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().
Referenced by do_compare(), do_hash(), and do_transform().

collate() [2/2]

```
template<typename _CharT>
std::__cxx11::collate< _CharT >::collate (
            __c_locale __cloc,
            size_t __refs = 0)  [inline], [explicit]
```

Internal constructor. Not for general use.
This is a constructor for use by the library itself to set up new locales.

Parameters

__cloc	The C locale.
__refs	Passed to the base facet class.

References std::locale::facet::facet().

~collate()

```
template<typename _CharT>
virtual std::__cxx11::collate< _CharT >::~collate ()  [inline], [protected], [virtual]
```

Destructor.

6.302.4 Member Function Documentation

compare()

```
template<typename _CharT>
int std::__cxx11::collate< _CharT >::compare (
            const _CharT * __lo1,
            const _CharT * __hi1,
            const _CharT * __lo2,
            const _CharT * __hi2) const  [inline]
```

Compare two strings.
This function compares two strings and returns the result by calling collate::do_compare().

Parameters

__lo1	Start of string 1.
__hi1	End of string 1.
__lo2	Start of string 2.
__hi2	End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

References do_compare().

do_compare()

```
template<typename _CharT>
int std::collate< _CharT >::do_compare (
            const _CharT * __lo1,
            const _CharT * __hi1,
            const _CharT * __lo2,
            const _CharT * __hi2) const  [protected], [virtual]
```

Compare two strings.
This function is a hook for derived classes to change the value returned.

See also

compare().

Parameters

__lo1	Start of string 1.
__hi1	End of string 1.
__lo2	Start of string 2.
__hi2	End of string 2.

Returns

1 if string1 > string2, -1 if string1 < string2, else 0.

References collate(), std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::data(), and std::basic_string< _CharT, _Traits, _Alloc >::length().
Referenced by compare().

do_hash()

```
template<typename _CharT>
long std::collate< _CharT >::do_hash (
            const _CharT * __lo,
            const _CharT * __hi) const  [protected], [virtual]
```

Return hash of a string.
This function computes and returns a hash on the input string. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _lo	Start of string.
_↩ _hi	End of string.

Returns

Hash value.

References collate().
Referenced by hash().

do_transform()

```
template<typename _CharT>
collate< _CharT >::string_type std::collate< _CharT >::do_transform (
            const _CharT * __lo,
            const _CharT * __hi) const  [protected], [virtual]
```

Transform string to comparable form.
This function is a hook for derived classes to change the value returned.

Parameters

_↩ _lo	Start.
_↩ _hi	End.

Returns

transformed string.

References collate(), std::basic_string< _CharT, _Traits, _Alloc >::append(), std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::basic_string< _CharT, _Traits, _Alloc >::clear(), std::basic_string< _CharT, _Traits, _Alloc >::data(), std::basic_string< _CharT, _Tra and std::basic_string< _CharT, _Traits, _Alloc >::push_back().
Referenced by transform().

hash()

```
template<typename _CharT>
long std::__cxx11::collate< _CharT >::hash (
            const _CharT * __lo,
            const _CharT * __hi) const  [inline]
```

Return hash of a string.
This function computes and returns a hash on the input string. It does so by returning collate::do_hash().

Parameters

_↩ _lo	Start of string.
_↩ _hi	End of string.

Returns

Hash value.

References do_hash().

transform()

```
template<typename _CharT>
string_type std::__cxx11::collate< _CharT >::transform (
            const _CharT * __lo,
            const _CharT * __hi) const  [inline]
```

Transform string to comparable form.

This function is a wrapper for strxfrm functionality. It takes the input string and returns a modified string that can be directly compared to other transformed strings. In the C locale, this function just returns a copy of the input string. In some other locales, it may replace two chars with one, change a char for another, etc. It does so by returning collate::do_transform().

Parameters

__lo	Start of string.
__hi	End of string.

Returns

Transformed string_type.

References do_transform().

6.302.5 Member Data Documentation

id

```
template<typename _CharT>
locale::id std::__cxx11::collate< _CharT >::id  [static]
```

Numpunct facet id.

The documentation for this class was generated from the following files:

- locale_classes.h
- locale_classes.tcc

6.303 std::__cxx11::collate_byname< _CharT > Class Template Reference

```
#include <locale_classes.h>
```

Public Types

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- **collate_byname** (const char *__s, size_t __refs=0)
- **collate_byname** (const string &__s, size_t __refs=0)
- int **_M_compare** (const char *, const char *) const throw()
- int **_M_compare** (const wchar_t *, const wchar_t *) const throw()
- size_t **_M_transform** (char *, const char *, size_t) const throw()
- size_t **_M_transform** (wchar_t *, const wchar_t *, size_t) const throw()

6.303.1 Detailed Description

template<typename _CharT>
class std::__cxx11::collate_byname< _CharT >

class collate_byname [22.2.4.2].

6.303.2 Member Typedef Documentation

char_type

```
template<typename _CharT>
typedef _CharT std::__cxx11::collate_byname< _CharT >::char_type
```

Public typedefs.

string_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::__cxx11::collate_byname< _CharT >::string_type
```

Public typedefs.
The documentation for this class was generated from the following file:

- locale_classes.h

6.304 std::common_iterator< _It, _Sent > Class Template Reference

```
#include <stl_iterator.h>
```

Public Member Functions

- constexpr **common_iterator** (_It __i) noexcept(is_nothrow_move_constructible_v< _It >)
- constexpr **common_iterator** (_Sent __s) noexcept(is_nothrow_move_constructible_v< _Sent >)
- **common_iterator** (common_iterator &&)=default
- constexpr **common_iterator** (common_iterator &&__x) noexcept(_S_noexcept< _It, _Sent >())
- **common_iterator** (const common_iterator &)=default
- constexpr **common_iterator** (const common_iterator &__x) noexcept(_S_noexcept< const _It &, const _Sent & >())
- template<typename _It2, typename _Sent2>
  constexpr **common_iterator** (const common_iterator< _It2, _Sent2 > &__x) noexcept(_S_noexcept< const _It2 &, const _Sent2 & >())
- constexpr decltype(auto) **operator*** ()
- constexpr decltype(auto) **operator*** () const
- constexpr common_iterator & **operator++** ()
- constexpr decltype(auto) **operator++** (int)
- constexpr auto **operator->** () const
- constexpr common_iterator & **operator=** (common_iterator &&)=default
- constexpr common_iterator & **operator=** (common_iterator &&__x) noexcept(is_nothrow_move_assignable_v< _It > &&is_nothrow_move_assignable_v< _Sent > &&is_nothrow_move_constructible_v< _It > &&is_nothrow_move_constructible_v< _Sent >)
- constexpr common_iterator & **operator=** (const common_iterator &)=default
- constexpr common_iterator & **operator=** (const common_iterator &__x) noexcept(is_nothrow_copy_assignable_v< _It > &&is_nothrow_copy_assignable_v< _Sent > &&is_nothrow_copy_constructible_v< _It > &&is_nothrow_copy_constructible_v< _Sent >)
- template<typename _It2, typename _Sent2>
  constexpr common_iterator & **operator=** (const common_iterator< _It2, _Sent2 > &__x) noexcept(is_nothrow_constructible_v< _It, const _It2 & > &&is_nothrow_constructible_v< _Sent, const _Sent2 & > &&is_nothrow_assignable_v< _It &, const _It2 & > &&is_nothrow_assignable_v< _Sent &, const _Sent2 & >)

Friends

- constexpr iter_rvalue_reference_t< _It > **iter_move** (const common_iterator &__i) noexcept(noexcept(ranges::iter_move(std::declval< const _It & >())))
- template<indirectly_swappable< _It > _It2, typename _Sent2>
  constexpr void **iter_swap** (const common_iterator &__x, const common_iterator< _It2, _Sent2 > &__y) noexcept(noexcept(ranges::iter_swap(std::declval< const _It & >(), std::declval< const _It2 & >())))
- template<sized_sentinel_for< _It > _It2, sized_sentinel_for< _It > _Sent2>
  constexpr iter_difference_t< _It2 > **operator-** (const common_iterator &__x, const common_iterator< _It2, _Sent2 > &__y)
- template<typename _It2, sentinel_for< _It > _Sent2>
  constexpr bool **operator==** (const common_iterator &__x, const common_iterator< _It2, _Sent2 > &__y)
- template<typename _It2, sentinel_for< _It > _Sent2>
  constexpr bool **operator==** (const common_iterator &__x, const common_iterator< _It2, _Sent2 > &__y)

6.304.1 Detailed Description

template<input_or_output_iterator _It, sentinel_for< _It > _Sent>
requires (!same_as<_It, _Sent>) && copyable<_It>
class std::common_iterator< _It, _Sent >

An iterator/sentinel adaptor for representing a non-common range.
The documentation for this class was generated from the following file:

- bits/stl_iterator.h

6.305 std::common_type< _Tp > Struct Template Reference

6.305.1 Detailed Description

template<typename... _Tp>
struct std::common_type< _Tp >

common_type
The documentation for this struct was generated from the following file:

- type_traits

6.306 std::common_type< chrono::duration< _Rep, _Period > > Struct Template Reference

`#include <chrono.h>`

Public Types

- using **type**

6.306.1 Detailed Description

template<typename _Rep, typename _Period>
struct std::common_type< chrono::duration< _Rep, _Period > >

Specialization of common_type for one chrono::duration type.
The documentation for this struct was generated from the following file:

- chrono.h

6.307 std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > > Struct Template Reference

`#include <chrono.h>`

Public Types

- using **type**

6.307.1 Detailed Description

template<typename _Rep, typename _Period>
struct std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >

Specialization of common_type for two identical chrono::duration types.
The documentation for this struct was generated from the following file:

- chrono.h

6.308 std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > > Struct Template Reference

`#include <chrono.h>`

6.308.1 Detailed Description

template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
struct std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >

Specialization of common_type for chrono::duration types.
The documentation for this struct was generated from the following file:

- chrono.h

6.309 std::common_type< chrono::time_point< _Clock, _Duration > > Struct Template Reference

`#include <chrono.h>`

Public Types

- using **type**

6.309.1 Detailed Description

template<typename _Clock, typename _Duration>
struct std::common_type< chrono::time_point< _Clock, _Duration > >

Specialization of common_type for one chrono::time_point type.
The documentation for this struct was generated from the following file:

- chrono.h

6.310 std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > > Struct Template Reference

`#include <chrono.h>`

Public Types

- using **type**

6.310.1 Detailed Description

template<typename _Clock, typename _Duration>
struct std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >

Specialization of common_type for two identical chrono::time_point types.
The documentation for this struct was generated from the following file:

- chrono.h

6.311 std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > > Struct Template Reference

`#include <chrono.h>`

6.311.1 Detailed Description

template<typename _Clock, typename _Duration1, typename _Duration2>
struct std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >

Specialization of common_type for chrono::time_point types.
The documentation for this struct was generated from the following file:

- chrono.h

6.312 std::compare_three_way_result< _Tp, _Up > Struct Template Reference

`#include <compare>`

6.312.1 Detailed Description

template<typename _Tp, typename _Up = _Tp>
struct std::compare_three_way_result< _Tp, _Up >

[cmp.result], result of three-way comparison
The documentation for this struct was generated from the following file:

- compare

6.313 std::complex< _Tp > Class Template Reference

`#include <complex>`

Public Types

- typedef _Tp value_type

Public Member Functions

- constexpr complex (const _Tp &__r=_Tp(), const _Tp &__i=_Tp())
- constexpr **complex** (const complex &)=default
- constexpr complex **__rep** () const
- template<typename _Up>
  explicit (!requires(_Up __u) { _Tp{__u};}) const expr complex(const complex< _Up > &__z)
- constexpr _Tp **imag** () const
- constexpr void **imag** (_Tp __val)
- constexpr complex< _Tp > & operator*= (const _Tp &)
- template<typename _Up>
  constexpr complex< _Tp > & operator*= (const complex< _Up > &)
- constexpr complex< _Tp > & operator+= (const _Tp &__t)
- template<typename _Up>
  constexpr complex< _Tp > & operator+= (const complex< _Up > &)
- constexpr complex< _Tp > & operator-= (const _Tp &__t)
- template<typename _Up>
  constexpr complex< _Tp > & operator-= (const complex< _Up > &)
- constexpr complex< _Tp > & operator/= (const _Tp &)
- template<typename _Up>
  constexpr complex< _Tp > & operator/= (const complex< _Up > &)
- constexpr complex< _Tp > & operator= (const _Tp &)
- constexpr complex & **operator=** (const complex &)=default
- template<typename _Up>
  constexpr complex< _Tp > & operator= (const complex< _Up > &)
- constexpr _Tp **real** () const
- constexpr void **real** (_Tp __val)

6.313.1 Detailed Description

template<typename _Tp>
class std::complex< _Tp >

Template to represent complex numbers.
Specializations for float, double, and long double are part of the library. Results with any other type are not guaranteed.

Parameters

Tp	Type of real and imaginary values.

6.313.2 Member Typedef Documentation

value_type

```
template<typename _Tp>
typedef _Tp std::complex< _Tp >::value_type
```

Value typedef.

6.313.3 Constructor & Destructor Documentation

complex()

```
template<typename _Tp>
std::complex< _Tp >::complex (
```

```
            const _Tp & __r = _Tp(),
            const _Tp & __i = _Tp())  [inline], [constexpr]
```

Default constructor. First parameter is x, second parameter is y. Unspecified parameters default to 0.
Referenced by operator*=(), operator*=(), operator+=(), operator-=(), operator/=(), operator/=(), operator=(), and operator=().

6.313.4 Member Function Documentation

explicit()

```
template<typename _Tp>
template<typename _Up>
std::complex< _Tp >::explicit (
            !requires(_Up __u) { _Tp{__u};} ) const &  [inline]
```

Converting constructor.

operator*=() [1/2]

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator*= (
            const _Tp & )  [constexpr]
```

Multiply this complex number by a scalar.
References complex().

operator*=() [2/2]

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator*= (
            const complex< _Up > & )  [constexpr]
```

Multiply this complex number by another.
References complex().

operator+=() [1/2]

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator+= (
            const _Tp & __t)  [inline], [constexpr]
```

Add a scalar to this complex number.

operator+=() [2/2]

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator+= (
            const complex< _Up > & )  [constexpr]
```

Add another complex number to this one.
References complex().

operator-=() [1/2]

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator-= (
            const _Tp & __t)  [inline], [constexpr]
```

Subtract a scalar from this complex number.

operator-=() [2/2]

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator-= (
            const complex< _Up > & )  [constexpr]
```

Subtract another complex number from this one.
References complex().

operator/=() [1/2]

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator/= (
            const _Tp & )  [constexpr]
```

Divide this complex number by a scalar.
References complex().

operator/=() [2/2]

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator/= (
            const complex< _Up > & )  [constexpr]
```

Divide this complex number by another.
References complex().

operator=() [1/2]

```
template<typename _Tp>
complex< _Tp > & std::complex< _Tp >::operator= (
            const _Tp & )  [constexpr]
```

Assign a scalar to this complex number.
References complex().

operator=() [2/2]

```
template<typename _Tp>
template<typename _Up>
complex< _Tp > & std::complex< _Tp >::operator= (
            const complex< _Up > & )  [constexpr]
```

Assign another complex number to this one.
References complex().
The documentation for this class was generated from the following file:

- complex

6.314 std::complex< double > Class Reference

```
#include <complex>
```

Public Types

- typedef __complex__ double **_ComplexT**
- typedef double **value_type**

Public Member Functions

- constexpr **complex** (_ComplexT __z)
- constexpr **complex** (const complex &)=default
- constexpr **complex** (double __r=0.0, double __i=0.0)
- **__attribute** ((__abi_tag__("cxx11"))) const expr double imag() const
- **__attribute** ((__abi_tag__("cxx11"))) const expr double real() const
- constexpr _ComplexT **__rep** () const
- template<typename _Up>
  explicit (!requires(_Up __u) { value_type{__u};}) const expr complex(const complex< _Up > &__z)
- constexpr void **imag** (double __val)
- template<typename _Tp>
  constexpr complex & **operator*=** (const complex< _Tp > &__z)
- constexpr complex & **operator*=** (double __d)
- template<typename _Tp>
  constexpr complex & **operator+=** (const complex< _Tp > &__z)
- constexpr complex & **operator+=** (double __d)
- template<typename _Tp>
  constexpr complex & **operator-=** (const complex< _Tp > &__z)
- constexpr complex & **operator-=** (double __d)
- template<typename _Tp>
  constexpr complex & **operator/=** (const complex< _Tp > &__z)
- constexpr complex & **operator/=** (double __d)
- constexpr complex & **operator=** (const complex &)=default
- template<typename _Tp>
  constexpr complex & **operator=** (const complex< _Tp > &__z)
- constexpr complex & **operator=** (double __d)
- constexpr void **real** (double __val)

6.314.1 Detailed Description

26.2.3 complex specializations complex<double> specialization
The documentation for this class was generated from the following file:

- complex

6.315 std::complex< float > Class Reference

`#include <complex>`

Public Types

- typedef __complex__ float **_ComplexT**
- typedef float **value_type**

Public Member Functions

- constexpr **complex** (_ComplexT __z)
- constexpr **complex** (const complex &)=default
- constexpr **complex** (float __r=0.0f, float __i=0.0f)
- **__attribute** ((__abi_tag__("cxx11"))) const expr float imag() const
- **__attribute** ((__abi_tag__("cxx11"))) const expr float real() const
- constexpr _ComplexT **__rep** () const

- template<typename _Up>
  explicit (!requires(_Up __u) { value_type{__u};}) const expr complex(const complex< _Up > &__z)
- constexpr void **imag** (float __val)
- template<class _Tp>
  constexpr complex & **operator*=** (const complex< _Tp > &__z)
- constexpr complex & **operator*=** (float __f)
- template<typename _Tp>
  constexpr complex & **operator+=** (const complex< _Tp > &__z)
- constexpr complex & **operator+=** (float __f)
- template<class _Tp>
  constexpr complex & **operator-=** (const complex< _Tp > &__z)
- constexpr complex & **operator-=** (float __f)
- template<class _Tp>
  constexpr complex & **operator/=** (const complex< _Tp > &__z)
- constexpr complex & **operator/=** (float __f)
- constexpr complex & **operator=** (const complex &)=default
- template<typename _Tp>
  constexpr complex & **operator=** (const complex< _Tp > &__z)
- constexpr complex & **operator=** (float __f)
- constexpr void **real** (float __val)

6.315.1 Detailed Description

26.2.3 complex specializations complex<float> specialization
The documentation for this class was generated from the following file:

- complex

6.316 std::complex< long double > Class Reference

`#include <complex>`

Public Types

- typedef __complex__ long double **_ComplexT**
- typedef long double **value_type**

Public Member Functions

- constexpr **complex** (_ComplexT __z)
- constexpr **complex** (const complex &)=default
- constexpr **complex** (long double __r=0.0L, long double __i=0.0L)
- **__attribute** ((__abi_tag__("cxx11"))) const expr long double imag() const
- **__attribute** ((__abi_tag__("cxx11"))) const expr long double real() const
- constexpr _ComplexT **__rep** () const
- template<typename _Up>
  explicit (!requires(_Up __u) { value_type{__u};}) const expr complex(const complex< _Up > &__z)
- constexpr void **imag** (long double __val)
- template<typename _Tp>
  constexpr complex & **operator*=** (const complex< _Tp > &__z)
- constexpr complex & **operator*=** (long double __r)
- template<typename _Tp>
  constexpr complex & **operator+=** (const complex< _Tp > &__z)

- constexpr complex & **operator+=** (long double __r)
- template<typename _Tp>
  constexpr complex & **operator-=** (const complex< _Tp > &__z)
- constexpr complex & **operator-=** (long double __r)
- template<typename _Tp>
  constexpr complex & **operator/=** (const complex< _Tp > &__z)
- constexpr complex & **operator/=** (long double __r)
- constexpr complex & **operator=** (const complex &)=default
- template<typename _Tp>
  constexpr complex & **operator=** (const complex< _Tp > &__z)
- constexpr complex & **operator=** (long double __r)
- constexpr void **real** (long double __val)

6.316.1 Detailed Description

26.2.3 complex specializations complex<long double> specialization
The documentation for this class was generated from the following file:

- complex

6.317 __gnu_pbds::detail::cond_dealtor< Entry, _Alloc > Class Template Reference

`#include <cond_key_dtor_entry_dealtor.hpp>`

Public Types

- typedef HT_Map::entry **entry**
- typedef HT_Map::entry_allocator **entry_allocator**
- typedef alloc_traits::allocator_type **entry_allocator**
- typedef alloc_traits::pointer **entry_pointer**
- typedef HT_Map::key_type **key_type**

Public Member Functions

- **cond_dealtor** (entry_allocator *p_a, entry *p_e)
- **cond_dealtor** (entry_pointer p_e)
- void **set_key_destruct** ()
- void **set_no_action** ()
- void **set_no_action_destructor** ()

Protected Attributes

- bool **m_key_destruct**
- entry_allocator *const **m_p_a**
- entry *const **m_p_e**

6.317.1 Detailed Description

template<typename Entry, typename _Alloc>
class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >

Conditional deallocate constructor argument.
Conditional dey destructor, cc_hash.
The documentation for this class was generated from the following files:

- cond_dealtor.hpp
- cond_key_dtor_entry_dealtor.hpp

6.318 __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type > Class Template Reference

`#include <ov_tree_map_.hpp>`

Public Member Functions

- **cond_dtor** (value_vector a_vec, iterator &r_last_it, Size_Type total_size)
- void **set_no_action** ()

Protected Attributes

- value_vector **m_a_vec**
- const Size_Type **m_max_size**
- bool **m_no_action**
- iterator & **m_r_last_it**

6.318.1 Detailed Description

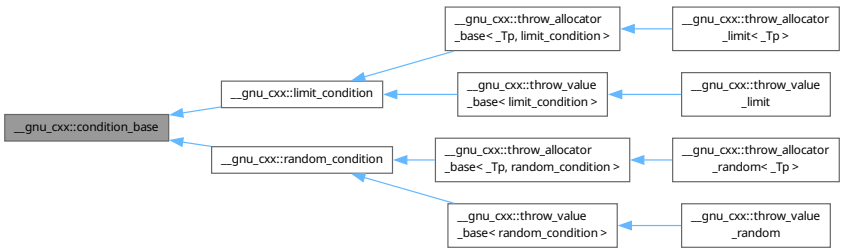
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
template<typename Size_Type>
class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >

Conditional destructor.
The documentation for this class was generated from the following file:

- ov_tree_map_.hpp

6.319 __gnu_cxx::condition_base Struct Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::condition_base:



Public Member Functions

- **condition_base** (const condition_base &)=default
- condition_base & **operator=** (const condition_base &)=default

6.319.1 Detailed Description

Base struct for condition policy.
Requires a public member function with the signature void throw_conditionally()
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.320 std::condition_variable Class Reference

`#include <condition_variable>`

Public Types

- typedef __gthread_cond_t * **native_handle_type**

Public Member Functions

- **condition_variable** (const condition_variable &)=delete
- native_handle_type **native_handle** ()
- void **notify_all** () noexcept
- void **notify_one** () noexcept
- condition_variable & **operator=** (const condition_variable &)=delete
- void **wait** (unique_lock< mutex > &__lock)
- template<typename _Predicate>
  void **wait** (unique_lock< mutex > &__lock, _Predicate __p)
- template<typename _Rep, typename _Period>
  cv_status **wait_for** (unique_lock< mutex > &__lock, chrono::duration< _Rep, _Period > __rtime)
- template<typename _Rep, typename _Period, typename _Predicate>
  bool **wait_for** (unique_lock< mutex > &__lock, chrono::duration< _Rep, _Period > __rtime, _Predicate __p)
- template<typename _Clock, typename _Duration>
  cv_status **wait_until** (unique_lock< mutex > &__lock, chrono::time_point< _Clock, _Duration > __atime)
- template<typename _Clock, typename _Duration, typename _Predicate>
  bool **wait_until** (unique_lock< mutex > &__lock, chrono::time_point< _Clock, _Duration > __atime, _Predicate __p)
- template<typename _Duration>
  cv_status **wait_until** (unique_lock< mutex > &__lock, chrono::time_point< steady_clock, _Duration > __atime)
- template<typename _Duration>
  cv_status **wait_until** (unique_lock< mutex > &__lock, chrono::time_point< system_clock, _Duration > __atime)

6.320.1 Detailed Description

condition_variable
The documentation for this class was generated from the following file:

- condition_variable

6.321 std::condition_variable_any Class Reference

`#include <condition_variable>`

Public Member Functions

- **condition_variable_any** (const condition_variable_any &)=delete
- void **notify_all** () noexcept
- void **notify_one** () noexcept
- condition_variable_any & **operator=** (const condition_variable_any &)=delete
- template<typename _Lock>
  void **wait** (_Lock &__lock)
- template<typename _Lock, typename _Predicate>
  void **wait** (_Lock &__lock, _Predicate __p)
- template<typename _Lock, typename _Rep, typename _Period>
  cv_status **wait_for** (_Lock &__lock, chrono::duration< _Rep, _Period > __rtime)
- template<typename _Lock, typename _Rep, typename _Period, typename _Predicate>
  bool **wait_for** (_Lock &__lock, chrono::duration< _Rep, _Period > __rtime, _Predicate __p)
- template<typename _Lock, typename _Clock, typename _Duration>
  cv_status **wait_until** (_Lock &__lock, chrono::time_point< _Clock, _Duration > __atime)
- template<typename _Lock, typename _Clock, typename _Duration, typename _Predicate>
  bool **wait_until** (_Lock &__lock, chrono::time_point< _Clock, _Duration > __atime, _Predicate __p)

6.321.1 Detailed Description

condition_variable_any
The documentation for this class was generated from the following file:

- condition_variable

6.322 std::conditional< _Cond, _Iftrue, _Iffalse > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.322.1 Detailed Description

template<bool _Cond, typename _Iftrue, typename _Iffalse>
struct std::conditional< _Cond, _Iftrue, _Iffalse >

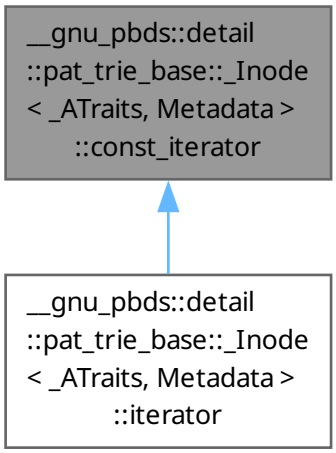
Define a member typedef `type` to one of two argument types.
The documentation for this struct was generated from the following file:

- type_traits

6.323 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator Struct Reference

`#include <pat_trie_base.hpp>`

Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator:



Public Types

- typedef _Alloc::difference_type **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef node_pointer_pointer **pointer**
- typedef node_pointer_reference **reference**
- typedef node_pointer **value_type**

Public Member Functions

- **const_iterator** (node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0)
- bool **operator!=** (const const_iterator &other) const
- node_const_pointer **operator*** () const
- const_iterator & **operator++** ()
- const_iterator **operator++** (int)
- const node_pointer_pointer **operator->** () const
- bool **operator==** (const const_iterator &other) const

Public Attributes

- node_pointer_pointer **m_p_p_cur**
- node_pointer_pointer **m_p_p_end**

6.323.1 Detailed Description

template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator

Constant child iterator.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

6.324 std::chrono::tzdb_list::const_iterator Class Reference

`#include <chrono>`

Public Types

- using **difference_type**
- using **iterator_category**
- using **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- **const_iterator** (const const_iterator &)=default
- **const_iterator** (const_iterator &&)=default
- reference **operator*** () const noexcept
- const_iterator & **operator++** ()
- const_iterator **operator++** (int)
- pointer **operator->** () const noexcept
- const_iterator & **operator=** (const const_iterator &)=default
- const_iterator & **operator=** (const_iterator &&)=default
- bool **operator==** (const const_iterator &) const noexcept=default

Friends

- class **tzdb_list**

6.324.1 Detailed Description

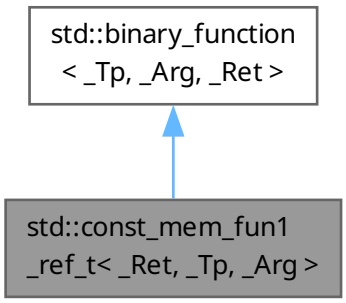
An iterator into the `tzdb_list`
As a extension, in libstdc++ each `tzdb` is reference-counted and the `const_iterator` type shares ownership of the object it refers to. This ensures that a `tzdb` erased from the list will not be destroyed while there is an iterator that refers to it.
The documentation for this class was generated from the following file:

- chrono

6.325 std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference

`#include <stl_function.h>`

Inheritance diagram for std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >:



Public Types

- typedef _Tp first_argument_type
- typedef _Ret result_type
- typedef _Arg second_argument_type

Public Member Functions

- **const_mem_fun1_ref_t** (_Ret(_Tp::*__pf)(_Arg) const)
- _Ret **operator()** (const _Tp &__r, _Arg __x) const

6.325.1 Detailed Description

template<typename _Ret, typename _Tp, typename _Arg>
class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >

One of the adaptors for member pointers.

6.325.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Arg, _Ret >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef _Ret std::binary_function< _Tp, _Arg, _Ret >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _Arg std::binary_function< _Tp, _Arg, _Ret >::second_argument_type  [inherited]
```

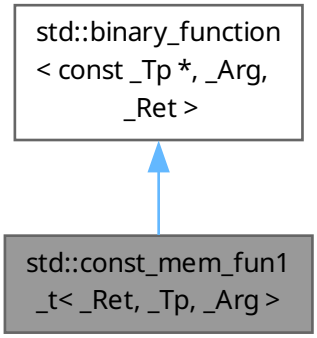
`second_argument_type` is the type of the second argument

The documentation for this class was generated from the following file:

- stl_function.h

6.326 std::const_mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::const_mem_fun1_t< _Ret, _Tp, _Arg >:



Public Types

- typedef const _Tp * first_argument_type
- typedef _Ret result_type
- typedef _Arg second_argument_type

Public Member Functions

- **const_mem_fun1_t** (_Ret(_Tp::*__pf)(_Arg) const)
- _Ret **operator()** (const _Tp *__p, _Arg __x) const

6.326.1 Detailed Description

template<typename _Ret, typename _Tp, typename _Arg>
class std::const_mem_fun1_t< _Ret, _Tp, _Arg >

One of the adaptors for member pointers.

6.326.2 Member Typedef Documentation

first_argument_type

`typedef const _Tp * std::binary_function< const _Tp *, _Arg, _Ret >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef _Ret std::binary_function< const _Tp *, _Arg, _Ret >::result_type [inherited]`
`result_type` is the return type

second_argument_type

```
typedef _Arg std::binary_function< const _Tp *, _Arg, _Ret >::second_argument_type [inherited]
```

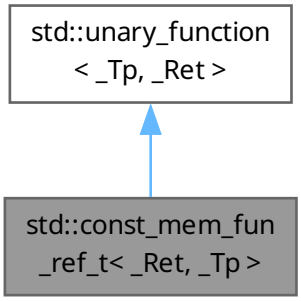
second_argument_type is the type of the second argument
The documentation for this class was generated from the following file:

- stl_function.h

6.327 std::const_mem_fun_ref_t< _Ret, _Tp > Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::const_mem_fun_ref_t< _Ret, _Tp >:



Public Types

- typedef _Tp argument_type
- typedef _Ret result_type

Public Member Functions

- **const_mem_fun_ref_t** (_Ret(_Tp::*__pf)() const)
- _Ret **operator()** (const _Tp &__r) const

6.327.1 Detailed Description

template<typename _Ret, typename _Tp>
class std::const_mem_fun_ref_t< _Ret, _Tp >

One of the adaptors for member pointers.

6.327.2 Member Typedef Documentation

argument_type

```
typedef _Tp std::unary_function< _Tp, _Ret >::argument_type [inherited]
```

argument_type is the type of the argument

result_type

```
typedef _Ret std::unary_function< _Tp, _Ret >::result_type  [inherited]
```

result_type is the return type

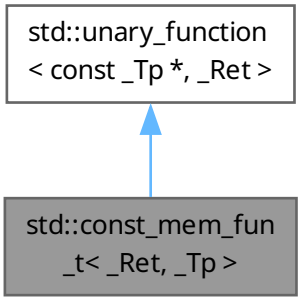
The documentation for this class was generated from the following file:

- stl_function.h

6.328 std::const_mem_fun_t< _Ret, _Tp > Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::const_mem_fun_t< _Ret, _Tp >:



Public Types

- typedef const _Tp * argument_type
- typedef _Ret result_type

Public Member Functions

- **const_mem_fun_t** (_Ret(_Tp::*__pf)() const)
- _Ret **operator()** (const _Tp *__p) const

6.328.1 Detailed Description

template<typename _Ret, typename _Tp>
class std::const_mem_fun_t< _Ret, _Tp >

One of the adaptors for member pointers.

6.328.2 Member Typedef Documentation

argument_type

```
typedef const _Tp * std::unary_function< const _Tp *, _Ret >::argument_type  [inherited]
```

argument_type is the type of the argument

result_type

```
typedef _Ret std::unary_function< const _Tp *, _Ret >::result_type  [inherited]
```

result_type is the return type

The documentation for this class was generated from the following file:

- stl_function.h

6.329 __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 > Struct Template Reference

```
#include <functional>
```

Public Types

- typedef _Arg1 **first_argument_type**
- typedef _Result **result_type**
- typedef _Arg2 **second_argument_type**

Public Member Functions

- **constant_binary_fun** (const _Result &__v)
- const result_type & **operator()** (const _Arg1 &, const _Arg2 &) const

Public Attributes

- _Result **_M_val**

6.329.1 Detailed Description

template<class _Result, class _Arg1 = _Result, class _Arg2 = _Arg1>
struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >

An SGI extension .

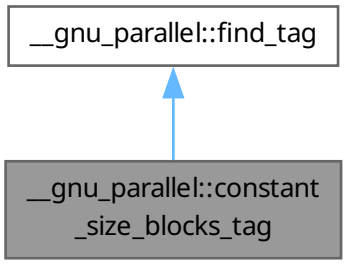
The documentation for this struct was generated from the following file:

- ext/functional

6.330 __gnu_parallel::constant_size_blocks_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::constant_size_blocks_tag:



6.330.1 Detailed Description

Selects the constant block size variant for std::find().

See also

_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS

The documentation for this struct was generated from the following file:

- tags.h

6.331 __gnu_cxx::constant_unary_fun< _Result, _Argument > Struct Template Reference

`#include <functional>`

Public Types

- typedef _Argument **argument_type**
- typedef _Result **result_type**

Public Member Functions

- **constant_unary_fun** (const _Result &__v)
- const result_type & **operator()** (const _Argument &) const

Public Attributes

- result_type **_M_val**

6.331.1 Detailed Description

template<class _Result, class _Argument = _Result>
struct __gnu_cxx::constant_unary_fun< _Result, _Argument >

An SGI extension .
The documentation for this struct was generated from the following file:

- ext/functional

6.332 __gnu_cxx::constant_void_fun< _Result > Struct Template Reference

`#include <functional>`

Public Types

- typedef _Result **result_type**

Public Member Functions

- **constant_void_fun** (const _Result &__v)
- const result_type & **operator()** () const

Public Attributes

- result_type **_M_val**

6.332.1 Detailed Description

template<class _Result>
struct __gnu_cxx::constant_void_fun< _Result >

An SGI extension .
The documentation for this struct was generated from the following file:

- ext/functional

6.333 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_Tl > Struct Template Reference

6.333.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Tag, typename Policy_Tl = null_type>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_Tl >

Dispatch mechanism, primary template for associative types.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.334 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type > Struct Template Reference

`#include <priority_queue_base_dispatch.hpp>`

Public Types

- typedef binary_heap< _VTp, Cmp_Fn, _Alloc > type

6.334.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >

Specialization for binary_heap.

6.334.2 Member Typedef Documentation

type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef binary_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp,
Cmp_Fn, _Alloc, binary_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

6.335 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

Public Types

- typedef binomial_heap< _VTp, Cmp_Fn, _Alloc > type

6.335.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >

Specialization for binomial_heap.

6.335.2 Member Typedef Documentation

type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp,
Cmp_Fn, _Alloc, binomial_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

6.336 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

Public Types

- typedef pairing_heap< _VTp, Cmp_Fn, _Alloc > type

6.336.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >

Specialization for pairing_heap.

6.336.2 Member Typedef Documentation

type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef pairing_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp,
Cmp_Fn, _Alloc, pairing_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

6.337 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

Public Types

- typedef rc_binomial_heap< _VTp, Cmp_Fn, _Alloc > type

6.337.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >

Specialization for rc_binary_heap.

6.337.2 Member Typedef Documentation

type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef rc_binomial_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp,
Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

6.338 __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type > Struct Template Reference

```
#include <priority_queue_base_dispatch.hpp>
```

Public Types

- typedef thin_heap< _VTp, Cmp_Fn, _Alloc > type

6.338.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >

Specialization for thin_heap.

6.338.2 Member Typedef Documentation

type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef thin_heap<_VTp, Cmp_Fn, _Alloc> __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_↩
Fn, _Alloc, thin_heap_tag, null_type >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- priority_queue_base_dispatch.hpp

6.339 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

Public Types

- typedef cc_ht_map< Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t > type

6.339.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >

Specialization collision-chaining hash map.

6.339.2 Member Typedef Documentation

type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef cc_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base_d
Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.340 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

Public Types

- typedef gp_ht_map< Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t > type

6.340.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >

Specialization general-probe hash map.

6.340.2 Member Typedef Documentation

type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef gp_ht_map<Key, Mapped, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::container_b
Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.341 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

Public Types

- typedef lu_map< Key, Mapped, at0t, _Alloc, at1t > type

6.341.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >

Specialization for list-update map.

6.341.2 Member Typedef Documentation

type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef lu_map<Key, Mapped, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch< Key,
Mapped, _Alloc, list_update_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.342 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

Public Types

- typedef ov_tree_map< Key, Mapped, at0t, at1t, _Alloc > type

6.342.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >

Specialization ordered-vector tree map.

6.342.2 Member Typedef Documentation

type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef ov_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.343 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

Public Types

- typedef pat_trie_map< Key, Mapped, at1t, _Alloc > **type**

6.343.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >

Specialization for PATRICIA trie map.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.344 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl > Struct Template Reference

```
#include <container_base_dispatch.hpp>
```

Public Types

- typedef rb_tree_map< Key, Mapped, at0t, at1t, _Alloc > type

6.344.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >

Specialization for R-B tree map.

6.344.2 Member Typedef Documentation

type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef rb_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >::type
```

Dispatched type.

The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.345 __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef splay_tree_map< Key, Mapped, at0t, at1t, _Alloc > type

6.345.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >

Specialization splay tree map.

6.345.2 Member Typedef Documentation

type

```
template<typename Key, typename Mapped, typename _Alloc, typename Policy_Tl>
typedef splay_tree_map<Key, Mapped, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.346 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef cc_ht_set< Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t > type

6.346.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >

Specialization collision-chaining hash set.

6.346.2 Member Typedef Documentation

type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef cc_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at2t> __gnu_pbds::detail::container_base
Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.347 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef gp_ht_set< Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t > type

6.347.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >

Specialization general-probe hash set.

6.347.2 Member Typedef Documentation

type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef gp_ht_set<Key, null_type, at0t, at1t, _Alloc, at3t::value, at4t, at5t, at2t> __gnu_pbds::detail::containe
Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.348 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef lu_set< Key, null_type, at0t, _Alloc, at1t > type

6.348.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >

Specialization for list-update set.

6.348.2 Member Typedef Documentation

type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef lu_set<Key, null_type, at0t, _Alloc, at1t> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, list_update_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.349 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef ov_tree_set< Key, null_type, at0t, at1t, _Alloc > type

6.349.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >

Specialization ordered-vector tree set.

6.349.2 Member Typedef Documentation

type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef ov_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.350 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef pat_trie_set< Key, null_type, at1t, _Alloc > type

6.350.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >

Specialization for PATRICIA trie set.

6.350.2 Member Typedef Documentation

type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef pat_trie_set<Key, null_type, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.351 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef rb_tree_set< Key, null_type, at0t, at1t, _Alloc > **type**

6.351.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >

Specialization for R-B tree set.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.352 __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl > Struct Template Reference

`#include <container_base_dispatch.hpp>`

Public Types

- typedef splay_tree_set< Key, null_type, at0t, at1t, _Alloc > type

6.352.1 Detailed Description

template<typename Key, typename _Alloc, typename Policy_Tl>
struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >

Specialization splay tree set.

6.352.2 Member Typedef Documentation

type

```
template<typename Key, typename _Alloc, typename Policy_Tl>
typedef splay_tree_set<Key, null_type, at0t, at1t, _Alloc> __gnu_pbds::detail::container_base_dispatch<
Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >::type
```

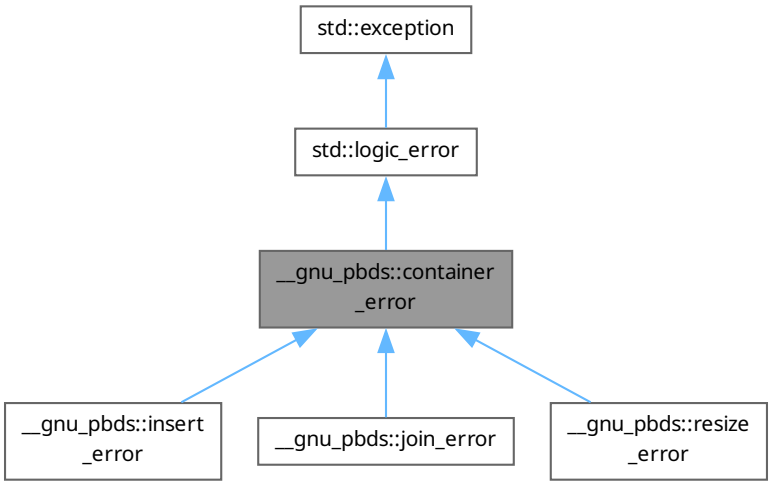
Dispatched type.
The documentation for this struct was generated from the following file:

- container_base_dispatch.hpp

6.353 __gnu_pbds::container_error Struct Reference

`#include <exception.hpp>`

Inheritance diagram for __gnu_pbds::container_error:



Public Member Functions

- virtual const char * what () const noexcept

6.353.1 Detailed Description

Base class for exceptions.

6.353.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
References what().
Referenced by what().
The documentation for this struct was generated from the following file:

- exception.hpp

6.354 __gnu_pbds::container_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::container_tag:



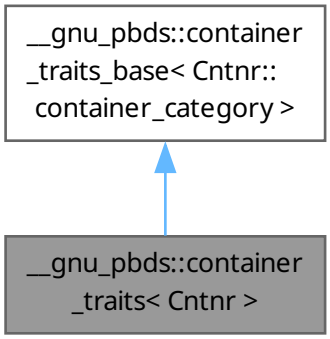
6.354.1 Detailed Description

Base data structure tag.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.355 __gnu_pbds::container_traits< Cntnr > Struct Template Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::container_traits< Cntnr >:



Public Types

- enum { order_preserving , erase_can_throw , split_join_can_throw , reverse_iteration }
- typedef container_traits_base< container_category > **base_type**
- typedef Cntnr::container_category **container_category**
- typedef Cntnr **container_type**
- typedef base_type::invalidation_guarantee **invalidation_guarantee**

6.355.1 Detailed Description

template<typename Cntnr>
struct __gnu_pbds::container_traits< Cntnr >

Container traits.

6.355.2 Member Enumeration Documentation

anonymous enum

```
template<typename Cntnr>
anonymous enum
```

Enumerator

order_preserving	True only if Cntnr objects guarantee storing keys by order.
erase_can_throw	True only if erasing a key can throw.
split_join_can_throw	True only if split or join operations can throw.
reverse_iteration	True only reverse iterators are supported.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.356 __gnu_pbds::container_traits_base< _Tag > Struct Template Reference

6.356.1 Detailed Description

template<typename _Tag>
struct __gnu_pbds::container_traits_base< _Tag >

Primary template, container traits base.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.357 __gnu_pbds::container_traits_base< binary_heap_tag > Struct Reference

```
#include <tag_and_trait.hpp>
```

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef binary_heap_tag **container_category**
- typedef basic_invalidation_guarantee **invalidation_guarantee**

6.357.1 Detailed Description

Specialization, binary heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.358 __gnu_pbds::container_traits_base< binomial_heap_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef binomial_heap_tag **container_category**
- typedef point_invalidation_guarantee **invalidation_guarantee**

6.358.1 Detailed Description

Specialization, binomial heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.359 __gnu_pbds::container_traits_base< cc_hash_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef cc_hash_tag **container_category**
- typedef point_invalidation_guarantee **invalidation_guarantee**

6.359.1 Detailed Description

Specialization, cc hash.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.360 __gnu_pbds::container_traits_base< gp_hash_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef gp_hash_tag **container_category**
- typedef basic_invalidation_guarantee **invalidation_guarantee**

6.360.1 Detailed Description

Specialization, gp hash.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.361 __gnu_pbds::container_traits_base< list_update_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef list_update_tag **container_category**
- typedef point_invalidation_guarantee **invalidation_guarantee**

6.361.1 Detailed Description

Specialization, list update.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.362 __gnu_pbds::container_traits_base< ov_tree_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef ov_tree_tag **container_category**
- typedef basic_invalidation_guarantee **invalidation_guarantee**

6.362.1 Detailed Description

Specialization, ov tree.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.363 __gnu_pbds::container_traits_base< pairing_heap_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef pairing_heap_tag **container_category**
- typedef point_invalidation_guarantee **invalidation_guarantee**

6.363.1 Detailed Description

Specialization, pairing heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.364 __gnu_pbds::container_traits_base< pat_trie_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef pat_trie_tag **container_category**
- typedef range_invalidation_guarantee **invalidation_guarantee**

6.364.1 Detailed Description

Specialization, pat trie.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.365 __gnu_pbds::container_traits_base< rb_tree_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef rb_tree_tag **container_category**
- typedef range_invalidation_guarantee **invalidation_guarantee**

6.365.1 Detailed Description

Specialization, rb tree.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.366 __gnu_pbds::container_traits_base< rc_binomial_heap_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef rc_binomial_heap_tag **container_category**
- typedef point_invalidation_guarantee **invalidation_guarantee**

6.366.1 Detailed Description

Specialization, rc binomial heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.367 __gnu_pbds::container_traits_base< splay_tree_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef splay_tree_tag **container_category**
- typedef range_invalidation_guarantee **invalidation_guarantee**

6.367.1 Detailed Description

Specialization, splay tree.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.368 __gnu_pbds::container_traits_base< thin_heap_tag > Struct Reference

`#include <tag_and_trait.hpp>`

Public Types

- enum { **order_preserving** , **erase_can_throw** , **split_join_can_throw** , **reverse_iteration** }
- typedef thin_heap_tag **container_category**
- typedef point_invalidation_guarantee **invalidation_guarantee**

6.368.1 Detailed Description

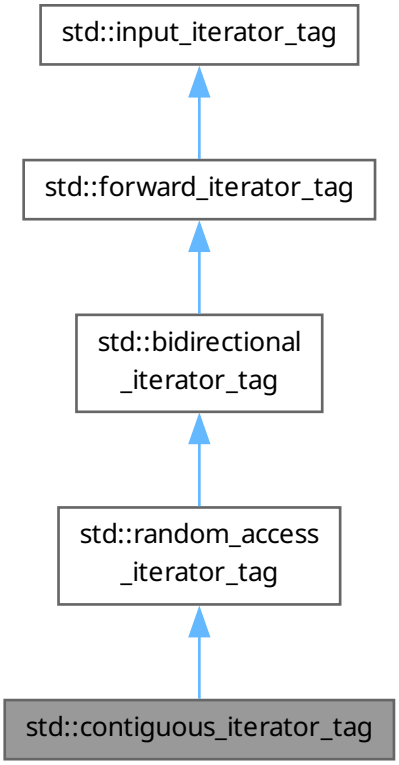
Specialization, thin heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.369 std::contiguous_iterator_tag Struct Reference

`#include <stl_iterator_base_types.h>`

Inheritance diagram for std::contiguous_iterator_tag:



6.369.1 Detailed Description

Contiguous iterators point to objects stored contiguously in memory.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

6.370 std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference

`#include <functional>`

Public Types

- using **result_type**

Public Member Functions

- copyable_function () noexcept
- template<typename _Fn, typename _Vt = decay_t<_Fn>>
  copyable_function (_Fn &&__f) noexcept(_S_nothrow_init< _Vt, _Fn >())

- copyable_function (copyable_function &&__x) noexcept
- copyable_function (copyable_function const &__x)
- template<typename _Tp, typename... _Args>
  copyable_function (in_place_type_t< _Tp >, _Args &&... __args) noexcept(_S_nothrow_init< _Tp, _Args... >())
- template<typename _Tp, typename _Up, typename... _Args>
  copyable_function (in_place_type_t< _Tp >, initializer_list< _Up > __il, _Args &&... __args) noexcept(_S_nothrow_init< _Tp, initializer_list< _Up > &, _Args... >())
- copyable_function (nullptr_t) noexcept
- operator bool () const noexcept
- _Res operator() (_ArgTypes... __args) _GLIBCXX_MOF_CV noexcept(_Noex)
- template<typename _Fn>
  copyable_function & operator= (_Fn &&__f) noexcept(is_nothrow_constructible_v< copyable_function, _Fn >)
- copyable_function & operator= (const copyable_function &__x)
- copyable_function & operator= (copyable_function &&__x) noexcept
- copyable_function & operator= (nullptr_t) noexcept
- void swap (copyable_function &__x) noexcept

Friends

- template<typename _Func>
  constexpr auto & **__polyfunc::__base_of** (_Func &) noexcept
- template<typename _Func>
  constexpr auto & **__polyfunc::__invoker_of** (_Func &) noexcept
- template<typename _Src, typename _Dst>
  consteval bool **__polyfunc::__is_invoker_convertible** () noexcept
- bool operator== (const copyable_function &__x, nullptr_t) noexcept
- void swap (copyable_function &__x, copyable_function &__y) noexcept

6.370.1 Detailed Description

template<typename _Res, typename... _ArgTypes, bool _Noex>
class std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

Polymorphic copyable function wrapper.

Since

C++26

The `std::copyable_function` class template is a call wrapper similar to `std::function`, but it does not provide information about it's target, and preserves constness.
It also supports const-qualification, ref-qualification, and no-throw guarantees. The qualifications and exception-specification of the `copyable_function::operator()` member function are respected when invoking the target function.

6.370.2 Constructor & Destructor Documentation

copyable_function() [1/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function
() [inline], [noexcept]
```

Creates an empty object.
References copyable_function().
Referenced by copyable_function(), copyable_function(), copyable_function(), copyable_function(), copyable_function(), copyable_function(), copyable_function(), operator=(), operator=(), operator=(), operator=(), operator==, swap(), and swap.

copyable_function() [2/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
            nullptr_t ) [inline], [noexcept]
```

Creates an empty object.
References copyable_function().

copyable_function() [3/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
            copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Moves the target object, leaving the source empty.
References copyable_function().

copyable_function() [4/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
            copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> const & __x)
[inline]
```

Copies the target object.
References copyable_function().

copyable_function() [5/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn, typename _Vt = decay_t<_Fn>>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
            _Fn && __f) [inline], [noexcept]
```

Stores a target object initialized from the argument.
References copyable_function(), and std::forward().

copyable_function() [6/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename... _Args>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
            in_place_type_t< _Tp > ,
            _Args &&... __args) [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.
References copyable_function(), and std::forward().

copyable_function() [7/7]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename _Up, typename... _Args>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::copyable_function (
            in_place_type_t< _Tp > ,
            initializer_list< _Up > __il,
            _Args &&... __args) [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.
References copyable_function(), and std::forward().

6.370.3 Member Function Documentation

operator bool()

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator bool ()
const [inline], [explicit], [noexcept]
```

True if a target object is present, false otherwise.

operator()()

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
_Res std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator() (
            _ArgTypes... __args) [inline], [noexcept]
```

Invoke the target object.

The target object will be invoked using the supplied arguments, and as an lvalue or rvalue, and as const or non-const, as dictated by the template arguments of the `copyable_function` specialization.

Precondition

Must not be empty.

References std::forward(), and operator()().

Referenced by operator()().

operator=() [1/4]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn>
copyable_function & std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_↩
Noex)>::operator= (
            _Fn && __f) [inline], [noexcept]
```

Stores a new target object, initialized from the argument.

References copyable_function(), std::forward(), and operator=().

operator=() [2/4]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
copyable_function & std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_↩
Noex)>::operator= (
            const copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x)
[inline]
```

Stores a copy of the source target object.

References copyable_function(), and operator=().

operator=() [3/4]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
copyable_function & std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_↩
Noex)>::operator= (
            copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Stores a new target object, leaving `x` empty.

References copyable_function(), std::addressof(), and operator=().

Referenced by operator=(), operator=(), operator=(), and operator=().

operator=() [4/4]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
copyable_function & std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_↩
Noex)>::operator= (
            nullptr_t ) [inline], [noexcept]
```

Destroys the target object (if any).
References copyable_function(), and operator=().

swap()

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::swap (
            copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x) [inline],
[noexcept]
```

Exchange the target objects (if any).
References copyable_function(), and swap().
Referenced by swap(), and swap.

6.370.4 Friends And Related Symbol Documentation

operator==

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
bool operator== (
            const copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x,
            nullptr_t ) [friend]
```

Check for emptiness by comparing with `nullptr`.
References copyable_function(), and operator==.
Referenced by operator==.

swap

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void swap (
            copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x,
            copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __y) [friend]
```

Exchange the target objects (if any).
References copyable_function(), and swap().
The documentation for this class was generated from the following file:

- cpyfunc_impl.h

6.371 std::counted_iterator< _It > Class Template Reference

`#include <stl_iterator.h>`

Public Types

- using **difference_type**
- using **iterator_type**

Public Member Functions

- constexpr **counted_iterator** (_It __i, iter_difference_t< _It > __n)

- template<typename _It2>
  constexpr **counted_iterator** (const counted_iterator< _It2 > &__x)
- constexpr _It **base** () &&noexcept(is_nothrow_move_constructible_v< _It >)
- constexpr const _It & **base** () const &noexcept
- constexpr iter_difference_t< _It > **count** () const noexcept
- constexpr decltype(auto) **operator*** () const noexcept(noexcept(*_M_current))
- constexpr decltype(auto) **operator*** () noexcept(noexcept(*_M_current))
- constexpr counted_iterator **operator+** (iter_difference_t< _It > __n) const
- constexpr counted_iterator & **operator++** ()
- constexpr counted_iterator **operator++** (int)
- constexpr decltype(auto) **operator++** (int)
- constexpr counted_iterator & **operator+=** (iter_difference_t< _It > __n)
- constexpr counted_iterator **operator-** (iter_difference_t< _It > __n) const
- constexpr counted_iterator & **operator--** ()
- constexpr counted_iterator **operator--** (int)
- constexpr counted_iterator & **operator-=** (iter_difference_t< _It > __n)
- constexpr auto **operator->** () const noexcept
- template<typename _It2>
  constexpr counted_iterator & **operator=** (const counted_iterator< _It2 > &__x)
- constexpr decltype(auto) **operator[ ]** (iter_difference_t< _It > __n) const noexcept(noexcept(_M_current[__n]))

Friends

- constexpr iter_rvalue_reference_t< _It > **iter_move** (const counted_iterator &__i) noexcept(noexcept(ranges::↩
  iter_move(__i._M_current)))
- template<indirectly_swappable< _It > _It2>
  constexpr void **iter_swap** (const counted_iterator &__x, const counted_iterator< _It2 > &__y) noexcept(noexcept(ranges::↩
  iter_swap(__x._M_current, __y._M_current)))
- constexpr counted_iterator **operator+** (iter_difference_t< _It > __n, const counted_iterator &__x)
- template<common_with< _It > _It2>
  constexpr iter_difference_t< _It2 > **operator-** (const counted_iterator &__x, const counted_iterator< _It2 > &↩
  __y) noexcept
- constexpr iter_difference_t< _It > **operator-** (const counted_iterator &__x, default_sentinel_t) noexcept
- constexpr iter_difference_t< _It > **operator-** (default_sentinel_t, const counted_iterator &__y) noexcept
- template<common_with< _It > _It2>
  constexpr strong_ordering **operator<=>** (const counted_iterator &__x, const counted_iterator< _It2 > &__y)
  noexcept
- template<common_with< _It > _It2>
  constexpr bool **operator==** (const counted_iterator &__x, const counted_iterator< _It2 > &__y) noexcept
- constexpr bool **operator==** (const counted_iterator &__x, default_sentinel_t) noexcept

6.371.1 Detailed Description

template<input_or_output_iterator _It>
class std::counted_iterator< _It >

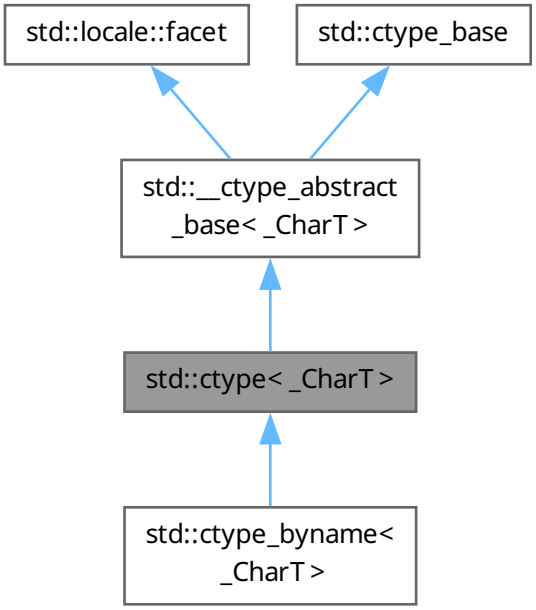
An iterator adaptor that keeps track of the distance to the end.
The documentation for this class was generated from the following file:

- bits/stl_iterator.h

6.372 std::ctype< _CharT > Class Template Reference

`#include <locale_facets.h>`

Inheritance diagram for std::ctype< _CharT >:



Public Types

- typedef const int * **__to_type**
- typedef _CharT **char_type**
- typedef __ctype_abstract_base< _CharT >::mask **mask**

Public Member Functions

- **ctype** (size_t __refs=0)
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- char_type widen (char __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- virtual bool do_is (mask __m, char_type __c) const
- virtual char do_narrow (char_type, char __dfault) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual char_type do_widen (char __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__dest) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.372.1 Detailed Description

template<typename _CharT>
class std::ctype< _CharT >

Primary class template ctype facet.
This template class defines classification and conversion functions for character sets. It wraps cctype functionality. Ctype gets used by streams for many I/O operations.
This template provides the protected virtual functions the developer will have to replace in a derived class or specialization to make a working facet. The public functions that access them are defined in __ctype_abstract_base, to allow for implementation flexibility. See ctype<wchar_t> for an example. The functions are documented in __ctype_abstract_↩ base.
Note: implementations are provided for all the protected virtual functions, but will likely not be useful.

6.372.2 Member Function Documentation

do_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [virtual]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_is() [2/2]

```
template<typename _CharT>
virtual bool std::ctype< _CharT >::do_is (
            mask __m,
            char_type __c) const  [protected], [virtual]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The char_type to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Implements std::__ctype_abstract_base< _CharT >.

do_narrow() [1/2]

```
template<typename _CharT>
virtual char std::ctype< _CharT >::do_narrow (
            char_type __c,
            char __dfault) const  [protected], [virtual]
```

Narrow char_type to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::narrow(), and std::ctype< char >::narrow().

do_narrow() [2/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [protected], [virtual]
```

Narrow char_type array to char.
This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_scan_is()

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *__hi*.

Implements std::__ctype_abstract_base< _CharT >.

do_scan_not()

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else *__hi*.

Implements std::__ctype_abstract_base< _CharT >.

do_tolower() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to lowercase.
This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_tolower() [2/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_tolower (
            char_type __c) const  [protected], [virtual]
```

Convert to lowercase.
This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::tolower(), and std::ctype< char >::tolower().

do_toupper() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to uppercase.
This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_toupper() [2/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_toupper (
            char_type __c) const  [protected], [virtual]
```

Convert to uppercase.
This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::toupper(), and std::ctype< char >::toupper().

do_widen() [1/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_widen (
            char __c) const  [protected], [virtual]
```

Widen char.

This virtual function converts the char to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::widen(), and std::ctype< char >::widen().

do_widen() [2/2]

```
template<typename _CharT>
virtual const char * std::ctype< _CharT >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [protected], [virtual]
```

Widen char array.
This function converts each char in the input to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

is() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [2/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
            mask __m,
            char_type __c) const  [inline], [inherited]
```

Test char_type classification.
This function finds a mask M for *__c* and compares it to mask *__m*. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _c	The char_type to compare the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

Referenced by std::time_get< _CharT, _InIter >::get().

narrow() [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Referenced by std::time_get< _CharT, _InIter >::do_get_year(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

narrow() [2/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline], [inherited]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

scan_is()

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else __*hi*.

scan_not()

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __*hi*.

tolower() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to lowercase.

This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

tolower() [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
            char_type __c) const  [inline], [inherited]
```

Convert to lowercase.
This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

__c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

Referenced by std::time_get< _CharT, _InIter >::get().

toupper() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

toupper() [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
            char_type __c) const  [inline], [inherited]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

Referenced by std::time_get< _CharT, _InIter >::get().

widen() [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::widen (
            char __c) const  [inline], [inherited]
```

Widen char to char_type.
This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::money_put< _CharT, _Out and std::time_put< _CharT, _OutIter >::do_put().

widen() [2/2]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline], [inherited]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

6.372.3 Member Data Documentation

id

```
template<typename _CharT>
locale::id std::ctype< _CharT >::id  [static]
```

The facet id for ctype<char_type>.

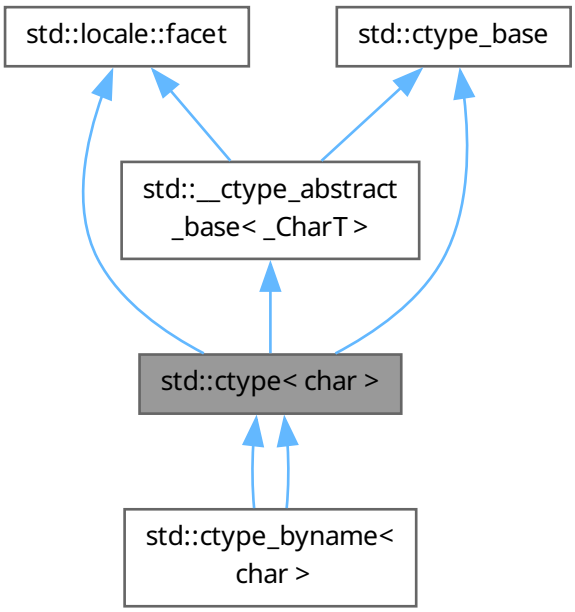
The documentation for this class was generated from the following file:

- locale_facets.h

6.373 std::ctype< char > Class Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::ctype< char >:



Public Types

- typedef const int * **__to_type**
- typedef char char_type
- typedef __ctype_abstract_base< char >::mask **mask**

Public Member Functions

- ctype (__c_locale __cloc, const mask *__table=0, bool __del=false, size_t __refs=0)
- ctype (const mask *__table=0, bool __del=false, size_t __refs=0)
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- const char * is (const char *__lo, const char *__hi, mask *__vec) const
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char __c) const
- bool is (mask __m, char_type __c) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- char narrow (char_type __c, char __dfault) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char * scan_is (mask __m, const char *__lo, const char *__hi) const

- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const char * scan_not (mask __m, const char *__lo, const char *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const mask * table () const throw ()
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- char_type tolower (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- char_type toupper (char_type __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const
- char_type widen (char __c) const
- char_type widen (char __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const

Static Public Member Functions

- static const mask * classic_table () throw ()

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size_t table_size
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual ~ctype ()
- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const =0
- virtual bool do_is (mask __m, char_type __c) const =0
- virtual bool do_is (mask __m, char_type __c) const
- virtual char do_narrow (char_type __c, char __dfault) const =0

- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const =0
- virtual char do_narrow (char_type __c, char __dfault) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const =0
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const =0
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const =0
- virtual char_type do_tolower (char_type __c) const =0
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const =0
- virtual char_type do_toupper (char_type __c) const =0
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__to) const =0
- virtual char_type do_widen (char __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__to) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_ctype**
- bool **_M_del**
- char **_M_narrow** [1+static_cast< unsigned char >(-1)]
- char **_M_narrow_ok**
- const mask * **_M_table**
- __to_type **_M_tolower**
- __to_type **_M_toupper**
- char **_M_widen** [1+static_cast< unsigned char >(-1)]
- char **_M_widen_ok**

6.373.1 Detailed Description

The ctype<char> specialization.
This class defines classification and conversion functions for the char type. It gets used by char streams for many I/O operations. The char specialization provides a number of optimizations as well.

6.373.2 Member Typedef Documentation

char_type

```
typedef char std::ctype< char >::char_type
```

Typedef for the template parameter char.

6.373.3 Constructor & Destructor Documentation

ctype() [1/2]

```
std::ctype< char >::ctype (
            const mask * __table = 0,
            bool __del = false,
            size_t __refs = 0)  [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__table	If non-zero, table is used as the per-char mask. Else classic_table() is used.
__del	If true, passes ownership of table to this facet.
__refs	Passed to the base facet class.

Referenced by ~ctype().

ctype() [2/2]

```
std::ctype< char >::ctype (
            __c_locale __cloc,
            const mask * __table = 0,
            bool __del = false,
            size_t __refs = 0)  [explicit]
```

Constructor performs static initialization.
This constructor is used to construct the initial C locale facet.

Parameters

__cloc	Handle to C locale data.
__table	If non-zero, table is used as the per-char mask.
__del	If true, passes ownership of table to this facet.
__refs	Passed to the base facet class.

~ctype()

```
virtual std::ctype< char >::~ctype ()  [protected], [virtual]
```

Destructor.
This function deletes table() if *del* was true in the constructor.
References ctype().

6.373.4 Member Function Documentation

classic_table()

```
const mask * std::ctype< char >::classic_table () throw ( )   [static]
```

Returns a pointer to the C locale mask table.

do_is() [1/3]

```
template<typename _CharT>
```

```
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [pure virtual], [inherited]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implemented in std::ctype< _CharT >, and std::mask< _CharT >.

do_is() [2/3]

```
template<typename _CharT>
virtual bool std::__ctype_abstract_base< _CharT >::do_is (
            mask __m,
            char_type __c) const  [protected], [pure virtual], [inherited]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The char_type to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::is(), and std::mask< _CharT >::is().

do_is() [3/3]

```
virtual bool std::ctype< char >::do_is (
            mask __m,
            char_type __c) const  [protected], [virtual]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The char_type to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

do_narrow() [1/4]

```
template<typename _CharT>
virtual char std::__ctype_abstract_base< _CharT >::do_narrow (
            char_type __c,
            char __dfault) const  [protected], [pure virtual], [inherited]
```

Narrow char_type to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::narrow(), and std::mask< _CharT >::narrow().

do_narrow() [2/4]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [protected], [pure virtual], [inherited]
```

Narrow char_type array to char.

This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implemented in std::ctype< _CharT >, and std::mask< _CharT >.

do_narrow() [3/4]

```
virtual char std::ctype< char >::do_narrow (
            char_type __c,
            char __dfault) const  [inline], [protected], [virtual]
```

Narrow char.
This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

do_narrow() [4/4]

```
virtual const char_type * std::ctype< char >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline], [protected], [virtual]
```

Narrow char array to char array.
This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

do_scan_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual], [inherited]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *__hi*.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::scan_is().

do_scan_is() [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _*m*	The mask to compare against.
_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else __*hi*.

do_scan_not() [1/2]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual], [inherited]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _*m*	The mask to compare against.
_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __*hi*.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::scan_not().

do_scan_not() [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else *__hi*.

do_tolower() [1/4]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual], [inherited]
```

Convert array to lowercase.
This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implemented in std::ctype< _CharT >, and std::mask< _CharT >.

do_tolower() [2/4]

```
template<typename _CharT>
```

```
virtual char_type std::__ctype_abstract_base< _CharT >::do_tolower (
            char_type __c) const  [protected], [pure virtual], [inherited]
```

Convert to lowercase.

This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, and std::mask< _CharT >.

Referenced by std::mask< _CharT >::tolower(), and std::mask< _CharT >::tolower().

do_tolower() [3/4]

```
virtual const char_type * std::ctype< char >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to lowercase.

This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

References do_tolower().

do_tolower() [4/4]

```
virtual char_type std::ctype< char >::do_tolower (
            char_type __c) const  [protected], [virtual]
```

Convert to lowercase.

This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.

do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The char to convert.

Returns

The lowercase char if convertible, else __c.

References do_tolower().
Referenced by do_tolower(), and do_tolower().

do_toupper() [1/4]

```
template<typename _CharT>
virtual const char_type * std::__ctype_abstract_base< _CharT >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [pure virtual], [inherited]
```

Convert array to uppercase.
This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implemented in std::ctype< _CharT >, and std::mask< _CharT >.

do_toupper() [2/4]

```
template<typename _CharT>
virtual char_type std::__ctype_abstract_base< _CharT >::do_toupper (
            char_type __c) const  [protected], [pure virtual], [inherited]
```

Convert to uppercase.
This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

Implemented in std::ctype< _CharT >, std::ctype< wchar_t >, and std::mask< _CharT >.
Referenced by std::mask< _CharT >::toupper(), and std::mask< _CharT >::toupper().

do_toupper() [3/4]

```
virtual const char_type * std::ctype< char >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to uppercase.
This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

References do_toupper().

do_toupper() [4/4]

```
virtual char_type std::ctype< char >::do_toupper (
            char_type __c) const  [protected], [virtual]
```

Convert to uppercase.
This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char to convert.

Returns

The uppercase char if convertible, else __c.

References do_toupper().
Referenced by do_toupper(), and do_toupper().

do_widen() [1/3]

```
template<typename _CharT>
virtual const char * std::__ctype_abstract_base< _CharT >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [protected], [pure virtual], [inherited]
```

Widen char array.
This function converts each char in the input to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

Implemented in std::ctype< _CharT >, and std::mask< _CharT >.

do_widen() [2/3]

```
virtual char_type std::ctype< char >::do_widen (
            char __c) const  [inline], [protected], [virtual]
```

Widen char.
This virtual function converts the char to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted character.

Implements std::__ctype_abstract_base< _CharT >.
References do_widen().
Referenced by do_widen().

do_widen() [3/3]

```
virtual const char * std::ctype< char >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline], [protected], [virtual]
```

Widen char array.

This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

is() [1/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [2/6]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
            mask __m,
            char_type __c) const  [inline], [inherited]
```

Test char_type classification.
This function finds a mask M for __c and compares it to mask __m. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _c	The char_type to compare the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

Referenced by std::time_get< _CharT, _InIter >::get().

is() [3/6]

```
const char * std::ctype< char >::is (
            const char * __lo,
            const char * __hi,
            mask * __vec) const  [inline]
```

Return a mask array.
This function finds the mask for each char in the range [lo, hi) and successively writes it to vec. vec must have as many elements as the char array.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [4/6]

```
const char_type * std::__ctype_abstract_base< char >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline]
```

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [5/6]

```
bool std::ctype< char >::is (
            mask __m,
            char __c) const  [inline]
```

Test char classification.
This function compares the mask table[c] to *__m*.

Parameters

_↩ _c	The char to compare the mask of.
_↩ _m	The mask to compare against.

Returns

True if __m & table[__c] is true, false otherwise.

is() [6/6]

```
bool std::__ctype_abstract_base< char >::is (
            mask __m,
            char_type __c) const  [inline]
```

Test char_type classification.
This function finds a mask M for *__c* and compares it to mask *__m*. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _c	The char_type to compare the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

narrow() [1/6]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Referenced by std::time_get< _CharT, _InIter >::do_get_year(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

narrow() [2/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline], [inherited]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

narrow() [3/6]

```
char std::ctype< char >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Narrow char.
This function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.
This function works as if it returns ctype<char>::do_narrow(c). do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.
__dfault	Char to return if conversion fails.

Returns

The converted character.

References std::ctype< _CharT >::do_narrow().

narrow() [4/6]

```
char std::__ctype_abstract_base< char >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

narrow() [5/6]

```
const char_type * std::ctype< char >::narrow (
            const char_type * __lo,
```

```
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline]
```

Narrow char array.

This function converts each char in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do_narrow(lo, hi, dfault, to). do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

References std::ctype< _CharT >::do_narrow().

narrow() [6/6]

```
const char_type * std::__ctype_abstract_base< char >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline]
```

Narrow array to char array.

This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

scan_is() [1/3]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else *__hi*.

scan_is() [2/3]

```
const char * std::ctype< char >::scan_is (
            mask __m,
            const char * __lo,
            const char * __hi) const  [inline]
```

Find char matching a mask.
This function searches for and returns the first char in [lo,hi) for which is(m,char) is true.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a matching char if found, else *__hi*.

scan_is() [3/3]

```
const char_type * std::__ctype_abstract_base< char >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else __hi.

scan_not() [1/3]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type not matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __hi.

scan_not() [2/3]

```
const char * std::ctype< char >::scan_not (
            mask __m,
```

```
            const char * __lo,
            const char * __hi) const  [inline]
```

Find char not matching a mask.
This function searches for and returns a pointer to the first char in [__lo,__hi) for which is(m,char) is false.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a non-matching char if found, else *__hi*.

scan_not() [3/3]

```
const char_type * std::__ctype_abstract_base< char >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type not matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else *__hi*.

table()

```
const mask * std::ctype< char >::table () const throw ( )   [inline]
```

Returns a pointer to the mask table provided to the constructor, or the default from classic_table() if none was provided.

tolower() [1/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
```

```
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to lowercase.

This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [2/6]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
            char_type __c) const  [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

Referenced by std::time_get< _CharT, _InIter >::get().

tolower() [3/6]

```
const char_type * std::ctype< char >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to lowercase.

This function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.
tolower() acts as if it returns ctype<char>:: do_tolower(__lo, __hi). do_tolower() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to first char in range.

_↩ _hi	Pointer to end of range.

Returns

__hi.

References std::ctype< _CharT >::do_tolower().

tolower() [4/6]

```
const char_type * std::__ctype_abstract_base< char >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to lowercase.
This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [5/6]

```
char_type std::ctype< char >::tolower (
            char_type __c) const  [inline]
```

Convert to lowercase.
This function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument. tolower() acts as if it returns ctype<char>::do_tolower(__c). do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The char to convert.

Returns

The lowercase char if convertible, else *__c*.

References std::ctype< _CharT >::do_tolower().

tolower() [6/6]

```
char_type std::__ctype_abstract_base< char >::tolower (
            char_type __c) const  [inline]
```

Convert to lowercase.
This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

toupper() [1/6]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [2/6]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
            char_type __c) const  [inline], [inherited]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

Referenced by std::time_get< _CharT, _InIter >::get().

toupper() [3/6]

```
const char_type * std::ctype< char >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to uppercase.
This function converts each char in the range [__lo,__hi) to uppercase if possible. Other chars remain untouched. toupper() acts as if it returns ctype<char>:: do_toupper(__lo, __hi). do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

References std::ctype< _CharT >::do_toupper().

toupper() [4/6]

```
const char_type * std::__ctype_abstract_base< char >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [5/6]

```
char_type std::ctype< char >::toupper (
            char_type __c) const  [inline]
```

Convert to uppercase.
This function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument. toupper() acts as if it returns ctype<char>::do_toupper(c). do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char to convert.

Returns

The uppercase char if convertible, else __c.

References std::ctype< _CharT >::do_toupper().

toupper() [6/6]

```
char_type std::__ctype_abstract_base< char >::toupper (
            char_type __c) const  [inline]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

widen() [1/5]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline], [inherited]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

_↩ _to	Pointer to the destination array.

Returns

__hi.

widen() [2/5]

```
char_type std::ctype< char >::widen (
            char __c) const  [inline]
```

Widen char.

This function converts the char to char_type using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be returned unchanged.

This function works as if it returns ctype<char>::do_widen(c). do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted character.

References std::ctype< _CharT >::do_widen().

widen() [3/5]

```
char_type std::__ctype_abstract_base< char >::widen (
            char __c) const  [inline]
```

Widen char to char_type.

This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type.

widen() [4/5]

```
const char * std::ctype< char >::widen (
            const char * __lo,
```

```
            const char * __hi,
            char_type * __to) const  [inline]
```

Widen char array.

This function converts each char in the input to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.

This function works as if it returns ctype<char>::do_widen(c). do_widen() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to first char in range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

References std::ctype< _CharT >::do_widen().

widen() [5/5]

```
const char * std::__ctype_abstract_base< char >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline]
```

Widen array to char_type.

This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

6.373.5 Member Data Documentation

id

```
locale::id std::ctype< char >::id  [static]
```

The facet id for ctype<char>.

table_size

```
const size_t std::ctype< char >::table_size  [static]
```

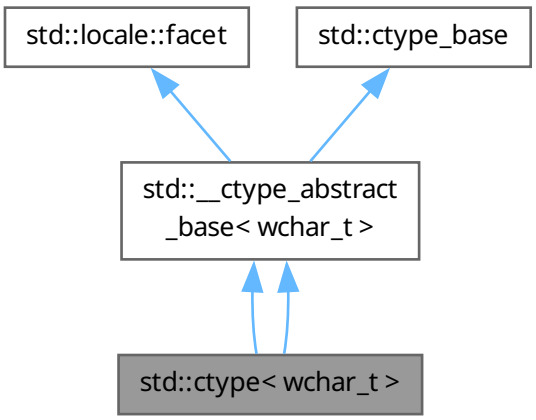
The size of the mask table. It is SCHAR_MAX + 1.
The documentation for this class was generated from the following files:

- locale_facets.h
- ctype_inline.h

6.374 std::ctype< wchar_t > Class Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::ctype< wchar_t >:



Public Types

- typedef const int * **__to_type**
- typedef wctype_t **__wmask_type**
- typedef wchar_t char_type
- typedef __ctype_abstract_base< wchar_t >::mask **mask**

Public Member Functions

- ctype (__c_locale __cloc, size_t __refs=0)
- ctype (size_t __refs=0)
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const

- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- char_type widen (char __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual ~ctype ()
- __wmask_type **_M_convert_to_wmask** (const mask __m) const throw ()
- void **_M_initialize_ctype** () throw ()
- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- virtual bool do_is (mask __m, char_type __c) const
- virtual char do_narrow (char_type __c, char __dfault) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual char_type do_widen (char __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__to) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- mask **_M_bit** [16]
- __c_locale **_M_c_locale_ctype**
- char **_M_narrow** [128]
- bool **_M_narrow_ok**
- wint_t **_M_widen** [1+static_cast< unsigned char >(-1)]
- __wmask_type **_M_wmask** [16]

6.374.1 Detailed Description

The ctype<wchar_t> specialization.
This class defines classification and conversion functions for the wchar_t type. It gets used by wchar_t streams for many I/O operations. The wchar_t specialization provides a number of optimizations as well.
ctype<wchar_t> inherits its public methods from __ctype_abstract_base<wchar_t>.

6.374.2 Member Typedef Documentation

char_type

```
typedef wchar_t std::ctype< wchar_t >::char_type
```

Typedef for the template parameter wchar_t.

6.374.3 Constructor & Destructor Documentation

ctype() [1/2]

```
std::ctype< wchar_t >::ctype (
            size_t __refs = 0)  [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

ctype() [2/2]

```
std::ctype< wchar_t >::ctype (
            __c_locale __cloc,
            size_t __refs = 0)  [explicit]
```

Constructor performs static initialization.
This constructor is used to construct the initial C locale facet.

Parameters

__cloc	Handle to C locale data.
__refs	Passed to the base facet class.

~ctype()

```
virtual std::ctype< wchar_t >::~ctype ()  [protected], [virtual]
```

Destructor.

6.374.4 Member Function Documentation

do_is() [1/2]

```
virtual const char_type * std::ctype< wchar_t >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [virtual]
```

Return a mask array.
This function finds the mask for each wchar_t in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

do_is() [2/2]

```
virtual bool std::ctype< wchar_t >::do_is (
            mask __m,
            char_type __c) const  [protected], [virtual]
```

Test wchar_t classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The wchar_t to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Implements std::__ctype_abstract_base< wchar_t >.

do_narrow() [1/2]

```
virtual char std::ctype< wchar_t >::do_narrow (
            char_type __c,
            char __dfault) const  [protected], [virtual]
```

Narrow wchar_t to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<wchar_t> facet, *c* will be cast to char and returned.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The wchar_t to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implements std::__ctype_abstract_base< wchar_t >.

do_narrow() [2/2]

```
virtual const char_type * std::ctype< wchar_t >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [protected], [virtual]
```

Narrow wchar_t array to char array.
This virtual function converts each wchar_t in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any wchar_t in the input that cannot be converted, *dfault* is used instead. For an underived ctype<wchar_t> facet, the argument will be copied, casting each element to char.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

do_scan_is()

```
virtual const char_type * std::ctype< wchar_t >::do_scan_is (
            mask __m,
```

```
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find wchar_t matching mask.
This function searches for and returns the first wchar_t c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a matching wchar_t if found, else __hi.

Implements std::__ctype_abstract_base< wchar_t >.

do_scan_not()

```
virtual const char_type * std::ctype< wchar_t >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find wchar_t not matching mask.
This function searches for and returns a pointer to the first wchar_t c of [__lo,__hi) for which is(__m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to a non-matching wchar_t if found, else __hi.

Implements std::__ctype_abstract_base< wchar_t >.

do_tolower() [1/2]

```
virtual const char_type * std::ctype< wchar_t >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to lowercase.
This virtual function converts each wchar_t in the range [lo,hi) to lowercase if possible. Other elements remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

__hi.

do_tolower() [2/2]

```
virtual char_type std::ctype< wchar_t >::do_tolower (
            char_type __c) const  [protected], [virtual]
```

Convert to lowercase.
This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

__c	The wchar_t to convert.

Returns

The lowercase wchar_t if convertible, else *__c*.

Implements std::__ctype_abstract_base< wchar_t >.

do_toupper() [1/2]

```
virtual const char_type * std::ctype< wchar_t >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to uppercase.
This virtual function converts each wchar_t in the range [lo,hi) to uppercase if possible. Other elements remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

do_toupper() [2/2]

```
virtual char_type std::ctype< wchar_t >::do_toupper (
            char_type __c) const  [protected], [virtual]
```

Convert to uppercase.
This virtual function converts the wchar_t argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The wchar_t to convert.

Returns

The uppercase wchar_t if convertible, else __c.

Implements std::__ctype_abstract_base< wchar_t >.

do_widen() [1/2]

```
virtual char_type std::ctype< wchar_t >::do_widen (
            char __c) const  [protected], [virtual]
```

Widen char to wchar_t.
This virtual function converts the char to wchar_t using the simplest reasonable transformation. For an underived ctype<wchar_t> facet, the argument will be cast to wchar_t.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted wchar_t.

Implements std::__ctype_abstract_base< wchar_t >.

do_widen() [2/2]

```
virtual const char * std::ctype< wchar_t >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [protected], [virtual]
```

Widen char array to wchar_t array.
This function converts each char in the input to wchar_t using the simplest reasonable transformation. For an underived ctype<wchar_t> facet, the argument will be copied, casting each element to wchar_t.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _*lo*	Pointer to start range.
_↩ _*hi*	Pointer to end of range.
_↩ _*to*	Pointer to the destination array.

Returns

__hi.

is() [1/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [2/2]

```
bool std::__ctype_abstract_base< wchar_t >::is (
```

```
            mask __m,
            char_type __c) const  [inline]
```

Test char_type classification.
This function finds a mask M for __c and compares it to mask __m. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _c	The char_type to compare the mask of.
_↩ _m	The mask to compare against.

Returns

(M & __m) != 0.

narrow() [1/2]

```
char std::__ctype_abstract_base< wchar_t >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

narrow() [2/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

scan_is()

```
const char_type * std::__ctype_abstract_base< wchar_t >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else *__hi*.

scan_not()

```
const char_type * std::__ctype_abstract_base< wchar_t >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __hi.

tolower() [1/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to lowercase.
This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [2/2]

```
char_type std::__ctype_abstract_base< wchar_t >::tolower (
            char_type __c) const  [inline]
```

Convert to lowercase.
This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

toupper() [1/2]

```
const char_type * std::__ctype_abstract_base< wchar_t >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [2/2]

```
char_type std::__ctype_abstract_base< wchar_t >::toupper (
            char_type __c) const  [inline]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

widen() [1/2]

```
char_type std::__ctype_abstract_base< wchar_t >::widen (
            char __c) const  [inline]
```

Widen char to char_type.
This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type.

widen() [2/2]

```
const char * std::__ctype_abstract_base< wchar_t >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

6.374.5 Member Data Documentation

id

```
locale::id std::ctype< wchar_t >::id  [static]
```

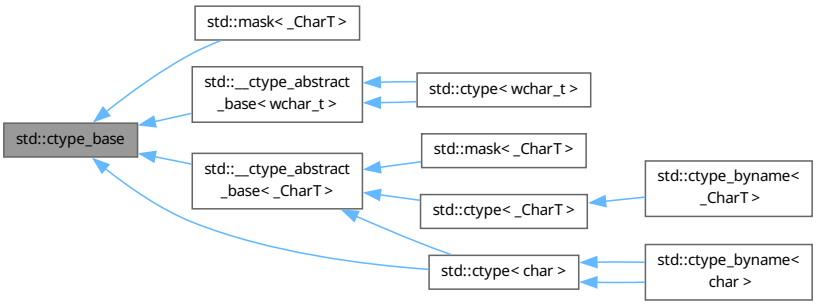
The facet id for ctype<wchar_t>.
The documentation for this class was generated from the following file:

- locale_facets.h

6.375 std::ctype_base Struct Reference

```
#include <ctype_base.h>
```

Inheritance diagram for std::ctype_base:



Public Types

- typedef const int * **__to_type**
- typedef unsigned short **mask**

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

6.375.1 Detailed Description

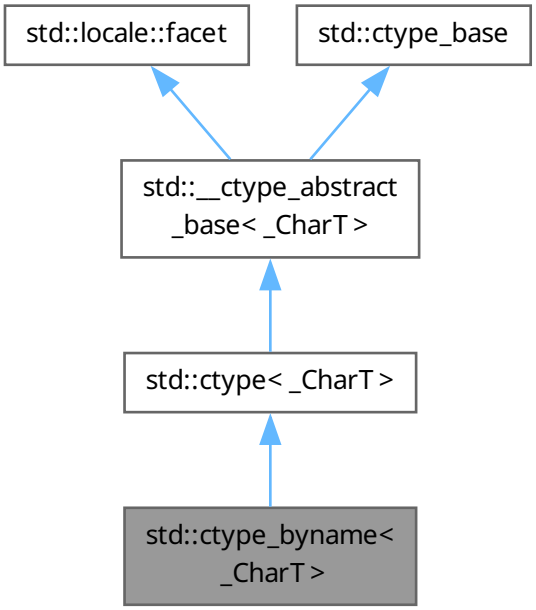
Base class for ctype.
The documentation for this struct was generated from the following file:

- ctype_base.h

6.376 std::ctype_byname< _CharT > Class Template Reference

`#include <locale_facets.h>`

Inheritance diagram for std::ctype_byname< _CharT >:



Public Types

- typedef const int * **__to_type**
- typedef _CharT **char_type**
- typedef ctype< _CharT >::mask **mask**

Public Member Functions

- **ctype_byname** (const char *__s, size_t __refs=0)
- **ctype_byname** (const string &__s, size_t __refs=0)
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- char_type widen (char __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- virtual bool do_is (mask __m, char_type __c) const
- virtual char do_narrow (char_type, char __dfault) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual char_type do_widen (char __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__dest) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.376.1 Detailed Description

template<typename _CharT>
class std::ctype_byname< _CharT >

class ctype_byname [22.2.1.2].

6.376.2 Member Function Documentation

do_is() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [virtual], [inherited]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_is() [2/2]

```
template<typename _CharT>
virtual bool std::ctype< _CharT >::do_is (
            mask __m,
            char_type __c) const  [protected], [virtual], [inherited]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__c	The char_type to find the mask of.
__m	The mask to compare against.

Returns

(M & __m) != 0.

Implements std::__ctype_abstract_base< _CharT >.

do_narrow() [1/2]

```
template<typename _CharT>
virtual char std::ctype< _CharT >::do_narrow (
            char_type __c,
            char __dfault) const  [protected], [virtual], [inherited]
```

Narrow char_type to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::narrow(), and std::ctype< char >::narrow().

do_narrow() [2/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [protected], [virtual], [inherited]
```

Narrow char_type array to char.
This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_scan_is()

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *__hi*.

Implements std::__ctype_abstract_base< _CharT >.

do_scan_not()

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __*hi*.

Implements std::__ctype_abstract_base< _CharT >.

do_tolower() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Convert array to lowercase.
This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

__*hi*.

Implements std::__ctype_abstract_base< _CharT >.

do_tolower() [2/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_tolower (
            char_type __c) const  [protected], [virtual], [inherited]
```

Convert to lowercase.
This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _*c*	The char_type to convert.

Returns

The lowercase char_type if convertible, else __*c*.

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::tolower(), and std::ctype< char >::tolower().

do_toupper() [1/2]

```
template<typename _CharT>
virtual const char_type * std::ctype< _CharT >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Convert array to uppercase.
This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

do_toupper() [2/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_toupper (
            char_type __c) const  [protected], [virtual], [inherited]
```

Convert to uppercase.
This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::toupper(), and std::ctype< char >::toupper().

do_widen() [1/2]

```
template<typename _CharT>
virtual char_type std::ctype< _CharT >::do_widen (
            char __c) const  [protected], [virtual], [inherited]
```

Widen char.

This virtual function converts the char to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type

Implements std::__ctype_abstract_base< _CharT >.
Referenced by std::ctype< char >::widen(), and std::ctype< char >::widen().

do_widen() [2/2]

```
template<typename _CharT>
virtual const char * std::ctype< _CharT >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [protected], [virtual], [inherited]
```

Widen char array.
This function converts each char in the input to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

Implements std::__ctype_abstract_base< _CharT >.

is() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline], [inherited]
```

Return a mask array.

This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [2/2]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
            mask __m,
            char_type __c) const  [inline], [inherited]
```

Test char_type classification.
This function finds a mask M for *__c* and compares it to mask *__m*. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _*c*	The char_type to compare the mask of.
_↩ _*m*	The mask to compare against.

Returns

(M & __m) != 0.

Referenced by std::time_get< _CharT, _InIter >::get().

narrow() [1/2]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

Referenced by std::time_get< _CharT, _InIter >::do_get_year(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

narrow() [2/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline], [inherited]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

scan_is()

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else __*hi*.

scan_not()

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type not matching a mask.

This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __*hi*.

tolower() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to lowercase.

This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
            char_type __c) const  [inline], [inherited]
```

Convert to lowercase.
This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

Referenced by std::time_get< _CharT, _InIter >::get().

toupper() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [2/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
            char_type __c) const  [inline], [inherited]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

Referenced by std::time_get< _CharT, _InIter >::get().

widen() [1/2]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::widen (
            char __c) const  [inline], [inherited]
```

Widen char to char_type.
This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type.

Referenced by std::money_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::money_put< _CharT, _Out and std::time_put< _CharT, _OutIter >::do_put().

widen() [2/2]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline], [inherited]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

6.376.3 Member Data Documentation

id

```
template<typename _CharT>
locale::id std::ctype< _CharT >::id  [static], [inherited]
```

The facet id for ctype<char_type>.

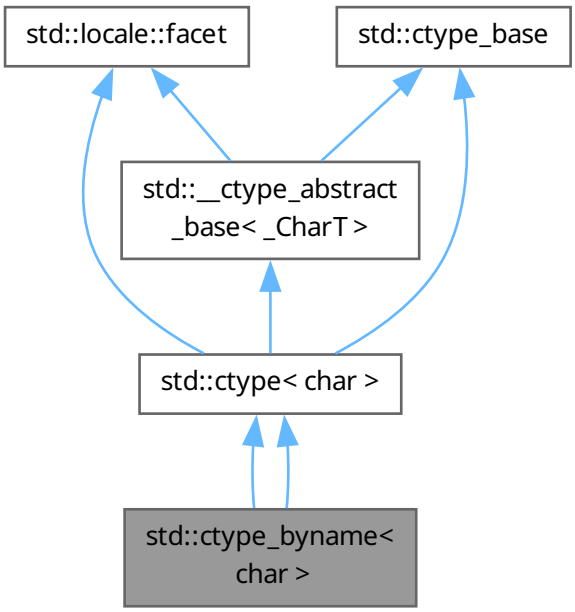
The documentation for this class was generated from the following file:

- locale_facets.h

6.377 std::ctype_byname< char > Class Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::ctype_byname< char >:



Public Types

- typedef const int * **__to_type**
- typedef char **char_type**
- typedef ctype< char >::mask **mask**

Public Member Functions

- **ctype_byname** (const char *__s, size_t __refs=0)
- **ctype_byname** (const string &__s, size_t __refs=0)
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- const char_type * is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- bool is (mask __m, char_type __c) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- char narrow (char_type __c, char __dfault) const
- const char_type * narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const char_type * scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- const mask * table () const throw ()
- const char_type * tolower (char_type *__lo, const char_type *__hi) const

- char_type tolower (char_type __c) const
- const char_type * tolower (char_type *__lo, const char_type *__hi) const
- char_type tolower (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- const char_type * toupper (char_type *__lo, const char_type *__hi) const
- char_type toupper (char_type __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const
- char_type widen (char __c) const
- const char * widen (const char *__lo, const char *__hi, char_type *__to) const

Static Public Member Functions

- static const mask * classic_table () throw ()

Static Public Attributes

- static const mask **alnum**
- static const mask **alpha**
- static const mask **blank**
- static const mask **cntrl**
- static const mask **digit**
- static const mask **graph**
- static locale::id id
- static const mask **lower**
- static const mask **print**
- static const mask **punct**
- static const mask **space**
- static const size_t table_size
- static const mask **upper**
- static const mask **xdigit**

Protected Member Functions

- virtual bool do_is (mask __m, char_type __c) const
- virtual const char_type * do_is (const char_type *__lo, const char_type *__hi, mask *__vec) const
- virtual bool do_is (mask __m, char_type __c) const
- virtual char do_narrow (char_type __c, char __dfault) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual char do_narrow (char_type, char __dfault) const
- virtual const char_type * do_narrow (const char_type *__lo, const char_type *__hi, char __dfault, char *__to) const
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_is (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_scan_not (mask __m, const char_type *__lo, const char_type *__hi) const
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type * do_tolower (char_type *__lo, const char_type *__hi) const
- virtual char_type do_tolower (char_type __c) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const

- virtual char_type do_toupper (char_type __c) const
- virtual const char_type * do_toupper (char_type *__lo, const char_type *__hi) const
- virtual char_type do_toupper (char_type __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__to) const
- virtual char_type do_widen (char __c) const
- virtual const char * do_widen (const char *__lo, const char *__hi, char_type *__dest) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_ctype**
- bool **_M_del**
- char **_M_narrow** [1+static_cast< unsigned char >(-1)]
- char **_M_narrow_ok**
- const mask * **_M_table**
- __to_type **_M_tolower**
- __to_type **_M_toupper**
- char **_M_widen** [1+static_cast< unsigned char >(-1)]
- char **_M_widen_ok**

6.377.1 Detailed Description

22.2.1.4 Class ctype_byname specializations.

6.377.2 Member Function Documentation

classic_table()

```
const mask * std::ctype< char >::classic_table () throw ( )   [static], [inherited]
```

Returns a pointer to the C locale mask table.

do_is() [1/3]

```
virtual bool std::ctype< char >::do_is (
            mask __m,
            char_type __c) const  [protected], [virtual], [inherited]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to find the mask of.

_↩ _*m*	The mask to compare against.

Returns

(M & __m) != 0.

do_is() [2/3]

```
virtual const char_type * std::ctype< char >::do_is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [protected], [virtual]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the input.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

do_is() [3/3]

```
virtual bool std::ctype< char >::do_is (
            mask __m,
            char_type __c) const  [protected], [virtual]
```

Test char_type classification.
This function finds a mask M for *c* and compares it to mask *m*.
do_is() is a hook for a derived facet to change the behavior of classifying. do_is() must always return the same result for the same input.

Parameters

_↩ _*c*	The char_type to find the mask of.
_↩ _*m*	The mask to compare against.

Returns

(M & __m) != 0.

do_narrow() [1/4]

```
virtual char std::ctype< char >::do_narrow (
            char_type __c,
            char __dfault) const  [inline], [protected], [virtual], [inherited]
```

Narrow char.

This virtual function converts the char to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. For an underived ctype<char> facet, *c* will be returned unchanged.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

do_narrow() [2/4]

```
virtual const char_type * std::ctype< char >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline], [protected], [virtual], [inherited]
```

Narrow char array to char array.

This virtual function converts each char in the range [lo,hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any char in the input that cannot be converted, *dfault* is used instead. For an underived ctype<char> facet, the argument will be copied unchanged.

do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.

Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

do_narrow() [3/4]

```
virtual char std::ctype< char >::do_narrow (
            char_type ,
            char __dfault) const  [protected], [virtual]
```

Narrow char_type to char.
This virtual function converts the argument to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

do_narrow() [4/4]

```
virtual const char_type * std::ctype< char >::do_narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [protected], [virtual]
```

Narrow char_type array to char.
This virtual function converts each char_type in the range [__lo,__hi) to char using the simplest reasonable transformation and writes the results to the destination array. For any element in the input that cannot be converted, *__dfault* is used instead.
do_narrow() is a hook for a derived facet to change the behavior of narrowing. do_narrow() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

do_scan_is() [1/2]

```
virtual const char_type * std::ctype< char >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *__hi*.

do_scan_is() [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type matching mask.
This function searches for and returns the first char_type c in [__lo,__hi) for which is(__m,c) is true.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

__m	The mask to compare against.
__lo	Pointer to start of range.
__hi	Pointer to end of range.

Returns

Pointer to a matching char_type if found, else *__hi*.

do_scan_not() [1/2]

```
virtual const char_type * std::ctype< char >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _*m*	The mask to compare against.
_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __*hi*.

do_scan_not() [2/2]

```
virtual const char_type * std::ctype< char >::do_scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Find char_type not matching mask.
This function searches for and returns a pointer to the first char_type c of [lo,hi) for which is(m,c) is false.
do_scan_is() is a hook for a derived facet to change the behavior of match searching. do_is() must always return the same result for the same input.

Parameters

_↩ _*m*	The mask to compare against.
_↩ _*lo*	Pointer to start of range.
_↩ _*hi*	Pointer to end of range.

Returns

Pointer to a non-matching char_type if found, else __*hi*.

do_tolower() [1/4]

```
virtual const char_type * std::ctype< char >::do_tolower (
```

```
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Convert array to lowercase.
This virtual function converts each char in the range [lo,hi) to lowercase if possible. Other chars remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

References do_tolower().

do_tolower() [2/4]

```
virtual char_type std::ctype< char >::do_tolower (
            char_type __c) const  [protected], [virtual], [inherited]
```

Convert to lowercase.
This virtual function converts the char argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The char to convert.

Returns

The lowercase char if convertible, else *__c*.

References do_tolower().
Referenced by do_tolower(), and do_tolower().

do_tolower() [3/4]

```
virtual const char_type * std::ctype< char >::do_tolower (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to lowercase.
This virtual function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

do_tolower() [4/4]

```
virtual char_type std::ctype< char >::do_tolower (
            char_type __c) const  [protected], [virtual]
```

Convert to lowercase.
This virtual function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument.
do_tolower() is a hook for a derived facet to change the behavior of lowercasing. do_tolower() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else *__c*.

do_toupper() [1/4]

```
virtual const char_type * std::ctype< char >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual], [inherited]
```

Convert array to uppercase.
This virtual function converts each char in the range [lo,hi) to uppercase if possible. Other chars remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

References do_toupper().

do_toupper() [2/4]

```
virtual char_type std::ctype< char >::do_toupper (
            char_type __c) const  [protected], [virtual], [inherited]
```

Convert to uppercase.
This virtual function converts the char argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char to convert.

Returns

The uppercase char if convertible, else __c.

References do_toupper().
Referenced by do_toupper(), and do_toupper().

do_toupper() [3/4]

```
virtual const char_type * std::ctype< char >::do_toupper (
            char_type * __lo,
            const char_type * __hi) const  [protected], [virtual]
```

Convert array to uppercase.
This virtual function converts each char_type in the range [__lo,__hi) to uppercase if possible. Other elements remain untouched.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

do_toupper() [4/4]

```
virtual char_type std::ctype< char >::do_toupper (
            char_type __c) const  [protected], [virtual]
```

Convert to uppercase.
This virtual function converts the char_type argument to uppercase if possible. If not possible (for example, '2'), returns the argument.
do_toupper() is a hook for a derived facet to change the behavior of uppercasing. do_toupper() must always return the same result for the same input.

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else __c.

do_widen() [1/3]

```
virtual const char * std::ctype< char >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline], [protected], [virtual], [inherited]
```

Widen char array.
This function converts each char in the range [lo,hi) to char using the simplest reasonable transformation. For an underived ctype<char> facet, the argument will be copied unchanged.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

do_widen() [2/3]

```
virtual char_type std::ctype< char >::do_widen (
            char __c) const  [protected], [virtual]
```

Widen char.
This virtual function converts the char to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type

do_widen() [3/3]

```
virtual const char * std::ctype< char >::do_widen (
            const char * __lo,
            const char * __hi,
            char_type * __dest) const  [protected], [virtual]
```

Widen char array.
This function converts each char in the input to char_type using the simplest reasonable transformation.
do_widen() is a hook for a derived facet to change the behavior of widening. do_widen() must always return the same result for the same input.
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start range.
__hi	Pointer to end of range.
__to	Pointer to the destination array.

Returns

__hi.

is() [1/4]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline], [inherited]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [2/4]

```
template<typename _CharT>
bool std::__ctype_abstract_base< _CharT >::is (
            mask __m,
            char_type __c) const  [inline], [inherited]
```

Test char_type classification.
This function finds a mask M for *__c* and compares it to mask *__m*. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _*c*	The char_type to compare the mask of.
_↩ _*m*	The mask to compare against.

Returns

(M & __m) != 0.

Referenced by std::time_get< _CharT, _InIter >::get().

is() [3/4]

```
const char_type * std::__ctype_abstract_base< char >::is (
            const char_type * __lo,
            const char_type * __hi,
            mask * __vec) const  [inline]
```

Return a mask array.
This function finds the mask for each char_type in the range [lo,hi) and successively writes it to vec. vec must have as many elements as the char array. It does so by returning the value of ctype<char_type>::do_is().

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__vec	Pointer to an array of mask storage.

Returns

__hi.

is() [4/4]

```
bool std::__ctype_abstract_base< char >::is (
            mask __m,
            char_type __c) const  [inline]
```

Test char_type classification.

This function finds a mask M for __*c* and compares it to mask __*m*. It does so by returning the value of ctype<char_↩ type>::do_is().

Parameters

_↩ _*c*	The char_type to compare the mask of.
_↩ _*m*	The mask to compare against.

Returns

(M & __m) != 0.

narrow() [1/4]

```
template<typename _CharT>
char std::__ctype_abstract_base< _CharT >::narrow (
            char_type __c,
            char __dfault) const  [inline], [inherited]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__*c*	The char_type to convert.
__*dfault*	Char to return if conversion fails.

Returns

The converted char.

Referenced by std::time_get< _CharT, _InIter >::do_get_year(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

narrow() [2/4]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline], [inherited]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

narrow() [3/4]

```
char std::__ctype_abstract_base< char >::narrow (
            char_type __c,
            char __dfault) const  [inline]
```

Narrow char_type to char.
This function converts the char_type to char using the simplest reasonable transformation. If the conversion fails, dfault is returned instead. It does so by returning ctype<char_type>::do_narrow(__c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__c	The char_type to convert.
__dfault	Char to return if conversion fails.

Returns

The converted char.

narrow() [4/4]

```
const char_type * std::__ctype_abstract_base< char >::narrow (
            const char_type * __lo,
            const char_type * __hi,
            char __dfault,
            char * __to) const  [inline]
```

Narrow array to char array.
This function converts each char_type in the input to char using the simplest reasonable transformation and writes the results to the destination array. For any char_type in the input that cannot be converted, *dfault* is used instead. It does so by returning ctype<char_type>::do_narrow(__lo, __hi, __dfault, __to).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

__lo	Pointer to start of range.
__hi	Pointer to end of range.
__dfault	Char to use if conversion fails.
__to	Pointer to the destination array.

Returns

__hi.

scan_is() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else *__hi*.

scan_is() [2/2]

```
const char_type * std::__ctype_abstract_base< char >::scan_is (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is true. It does so by returning ctype<char_type>::do_scan_is().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to matching char_type if found, else __*hi*.

scan_not() [1/2]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Find char_type not matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else __*hi*.

scan_not() [2/2]

```
const char_type * std::__ctype_abstract_base< char >::scan_not (
            mask __m,
            const char_type * __lo,
            const char_type * __hi) const  [inline]
```

Find char_type not matching a mask.
This function searches for and returns the first char_type c in [lo,hi) for which is(m,c) is false. It does so by returning ctype<char_type>::do_scan_not().

Parameters

_↩ _m	The mask to compare against.
_↩ _lo	Pointer to first char in range.
_↩ _hi	Pointer to end of range.

Returns

Pointer to non-matching char if found, else *__hi*.

table()

```
const mask * std::ctype< char >::table () const throw ( )   [inline], [inherited]
```

Returns a pointer to the mask table provided to the constructor, or the default from classic_table() if none was provided.

tolower() [1/4]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to lowercase.

This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [2/4]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::tolower (
            char_type __c) const  [inline], [inherited]
```

Convert to lowercase.

This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

Referenced by std::time_get< _CharT, _InIter >::get().

tolower() [3/4]

```
const char_type * std::__ctype_abstract_base< char >::tolower (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to lowercase.
This function converts each char_type in the range [__lo,__hi) to lowercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_tolower(__lo, __hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

tolower() [4/4]

```
char_type std::__ctype_abstract_base< char >::tolower (
            char_type __c) const  [inline]
```

Convert to lowercase.
This function converts the argument to lowercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_tolower(c).

Parameters

_↩ _c	The char_type to convert.

Returns

The lowercase char_type if convertible, else __c.

toupper() [1/4]

```
template<typename _CharT>
const char_type * std::__ctype_abstract_base< _CharT >::toupper (
```

```
            char_type * __lo,
            const char_type * __hi) const  [inline], [inherited]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [2/4]

```
template<typename _CharT>
char_type std::__ctype_abstract_base< _CharT >::toupper (
            char_type __c) const  [inline], [inherited]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

Referenced by std::time_get< _CharT, _InIter >::get().

toupper() [3/4]

```
const char_type * std::__ctype_abstract_base< char >::toupper (
            char_type * __lo,
            const char_type * __hi) const  [inline]
```

Convert array to uppercase.
This function converts each char_type in the range [lo,hi) to uppercase if possible. Other elements remain untouched. It does so by returning ctype<char_type>:: do_toupper(lo, hi).

Parameters

_↩ _lo	Pointer to start of range.

_↩ _hi	Pointer to end of range.

Returns

__hi.

toupper() [4/4]

```
char_type std::__ctype_abstract_base< char >::toupper (
            char_type __c) const  [inline]
```

Convert to uppercase.
This function converts the argument to uppercase if possible. If not possible (for example, '2'), returns the argument. It does so by returning ctype<char_type>::do_toupper().

Parameters

_↩ _c	The char_type to convert.

Returns

The uppercase char_type if convertible, else *__c*.

widen() [1/3]

```
template<typename _CharT>
const char * std::__ctype_abstract_base< _CharT >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline], [inherited]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

widen() [2/3]

```
char_type std::__ctype_abstract_base< char >::widen (
            char __c) const  [inline]
```

Widen char to char_type.
This function converts the char argument to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _c	The char to convert.

Returns

The converted char_type.

widen() [3/3]

```
const char * std::__ctype_abstract_base< char >::widen (
            const char * __lo,
            const char * __hi,
            char_type * __to) const  [inline]
```

Widen array to char_type.
This function converts each char in the input to char_type using the simplest reasonable transformation. It does so by returning ctype<char_type>::do_widen(c).
Note: this is not what you want for codepage conversions. See codecvt for that.

Parameters

_↩ _lo	Pointer to start of range.
_↩ _hi	Pointer to end of range.
_↩ _to	Pointer to the destination array.

Returns

__hi.

6.377.3 Member Data Documentation

id

```
locale::id std::ctype< char >::id  [static]
```

The facet id for ctype<char_type>.

table_size

```
const size_t std::ctype< char >::table_size  [static], [inherited]
```

The size of the mask table. It is SCHAR_MAX + 1.
The documentation for this class was generated from the following file:

- locale_facets.h

6.378 std::ranges::dangling Struct Reference

`#include <ranges_base.h>`

Public Member Functions

- template<typename... _Args>
  constexpr **dangling** (_Args &&...) noexcept

6.378.1 Detailed Description

Type returned by algorithms instead of a dangling iterator or subrange.
The documentation for this struct was generated from the following file:

- ranges_base.h

6.379 __gnu_cxx::debug_allocator< _Alloc > Class Template Reference

`#include <debug_allocator.h>`

Public Types

- typedef _Traits::const_pointer **const_pointer**
- typedef _Traits::const_reference **const_reference**
- typedef _Traits::difference_type **difference_type**
- typedef _Traits::pointer **pointer**
- typedef _Traits::reference **reference**
- typedef _Traits::size_type **size_type**
- typedef _Traits::value_type **value_type**

Public Member Functions

- **debug_allocator** (const _Alloc &__a)
- template<typename _Alloc2>
  debug_allocator (const debug_allocator< _Alloc2 > &__a2, typename __convertible< _Alloc2 >::__type=0)
- pointer **allocate** (size_type __n)
- pointer **allocate** (size_type __n, const void *__hint)
- template<typename _Tp, typename... _Args>
  void **construct** (_Tp *__p, _Args &&... __args)
- void **construct** (pointer __p, const value_type &__val)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Tp>
  void **destroy** (_Tp *__p)
- size_type **max_size** () const throw ()

Friends

- template<typename _Alloc2>
  bool **operator==** (const debug_allocator &__lhs, const debug_allocator< _Alloc2 > &__rhs) noexcept

6.379.1 Detailed Description

template<**typename _Alloc**>
class __gnu_cxx::debug_allocator< **_Alloc** >

A meta-allocator with debugging bits.
This is precisely the allocator defined in the C++03 Standard.
The documentation for this class was generated from the following file:

- debug_allocator.h

6.380 std::decimal::decimal128 Class Reference

`#include <decimal>`

Public Types

- typedef float **__decfloat128**

Public Member Functions

- decimal128 (__decfloat128 __z)
- **decimal128** (decimal32 d32)
- **decimal128** (decimal64 d64)
- **decimal128** (double __r)
- **decimal128** (float __r)
- **decimal128** (int __z)
- **decimal128** (long __z)
- **decimal128** (long double __r)
- **decimal128** (long long __z)
- **decimal128** (unsigned int __z)
- **decimal128** (unsigned long __z)
- **decimal128** (unsigned long long __z)
- __decfloat128 **__getval** (void)
- void **__setval** (__decfloat128 __x)
- **operator long long** () const
- decimal128 & **operator*=** (decimal128 __rhs)
- decimal128 & **operator*=** (decimal32 __rhs)
- decimal128 & **operator*=** (decimal64 __rhs)
- decimal128 & **operator*=** (int __rhs)
- decimal128 & **operator*=** (long __rhs)
- decimal128 & **operator*=** (long long __rhs)
- decimal128 & **operator*=** (unsigned int __rhs)
- decimal128 & **operator*=** (unsigned long __rhs)
- decimal128 & **operator*=** (unsigned long long __rhs)
- decimal128 & **operator++** ()
- decimal128 **operator++** (int)
- decimal128 & **operator+=** (decimal128 __rhs)
- decimal128 & **operator+=** (decimal32 __rhs)
- decimal128 & **operator+=** (decimal64 __rhs)
- decimal128 & **operator+=** (int __rhs)
- decimal128 & **operator+=** (long __rhs)
- decimal128 & **operator+=** (long long __rhs)

- decimal128 & **operator+=** (unsigned int __rhs)
- decimal128 & **operator+=** (unsigned long __rhs)
- decimal128 & **operator+=** (unsigned long long __rhs)
- decimal128 & **operator--** ()
- decimal128 **operator--** (int)
- decimal128 & **operator-=** (decimal128 __rhs)
- decimal128 & **operator-=** (decimal32 __rhs)
- decimal128 & **operator-=** (decimal64 __rhs)
- decimal128 & **operator-=** (int __rhs)
- decimal128 & **operator-=** (long __rhs)
- decimal128 & **operator-=** (long long __rhs)
- decimal128 & **operator-=** (unsigned int __rhs)
- decimal128 & **operator-=** (unsigned long __rhs)
- decimal128 & **operator-=** (unsigned long long __rhs)
- decimal128 & **operator/=** (decimal128 __rhs)
- decimal128 & **operator/=** (decimal32 __rhs)
- decimal128 & **operator/=** (decimal64 __rhs)
- decimal128 & **operator/=** (int __rhs)
- decimal128 & **operator/=** (long __rhs)
- decimal128 & **operator/=** (long long __rhs)
- decimal128 & **operator/=** (unsigned int __rhs)
- decimal128 & **operator/=** (unsigned long __rhs)
- decimal128 & **operator/=** (unsigned long long __rhs)

6.380.1 Detailed Description

3.2.4 Class decimal128.

6.380.2 Constructor & Destructor Documentation

decimal128()

```
std::decimal::decimal128::decimal128 (
            __decfloat128 __z)  [inline]
```

Conforming extension: Conversion from scalar decimal type.
The documentation for this class was generated from the following file:

- decimal

6.381 std::decimal::decimal32 Class Reference

```
#include <decimal>
```

Public Types

- typedef float **__decfloat32**

Public Member Functions

- decimal32 (__decfloat32 __z)
- **decimal32** (decimal128 __d128)
- **decimal32** (decimal64 __d64)
- **decimal32** (double __r)
- **decimal32** (float __r)
- **decimal32** (int __z)
- **decimal32** (long __z)
- **decimal32** (long double __r)
- **decimal32** (long long __z)
- **decimal32** (unsigned int __z)
- **decimal32** (unsigned long __z)
- **decimal32** (unsigned long long __z)
- __decfloat32 **__getval** (void)
- void **__setval** (__decfloat32 __x)
- **operator long long** () const
- decimal32 & **operator*=** (decimal128 __rhs)
- decimal32 & **operator*=** (decimal32 __rhs)
- decimal32 & **operator*=** (decimal64 __rhs)
- decimal32 & **operator*=** (int __rhs)
- decimal32 & **operator*=** (long __rhs)
- decimal32 & **operator*=** (long long __rhs)
- decimal32 & **operator*=** (unsigned int __rhs)
- decimal32 & **operator*=** (unsigned long __rhs)
- decimal32 & **operator*=** (unsigned long long __rhs)
- decimal32 & **operator++** ()
- decimal32 **operator++** (int)
- decimal32 & **operator+=** (decimal128 __rhs)
- decimal32 & **operator+=** (decimal32 __rhs)
- decimal32 & **operator+=** (decimal64 __rhs)
- decimal32 & **operator+=** (int __rhs)
- decimal32 & **operator+=** (long __rhs)
- decimal32 & **operator+=** (long long __rhs)
- decimal32 & **operator+=** (unsigned int __rhs)
- decimal32 & **operator+=** (unsigned long __rhs)
- decimal32 & **operator+=** (unsigned long long __rhs)
- decimal32 & **operator--** ()
- decimal32 **operator--** (int)
- decimal32 & **operator-=** (decimal128 __rhs)
- decimal32 & **operator-=** (decimal32 __rhs)
- decimal32 & **operator-=** (decimal64 __rhs)
- decimal32 & **operator-=** (int __rhs)
- decimal32 & **operator-=** (long __rhs)
- decimal32 & **operator-=** (long long __rhs)
- decimal32 & **operator-=** (unsigned int __rhs)
- decimal32 & **operator-=** (unsigned long __rhs)
- decimal32 & **operator-=** (unsigned long long __rhs)
- decimal32 & **operator/=** (decimal128 __rhs)
- decimal32 & **operator/=** (decimal32 __rhs)
- decimal32 & **operator/=** (decimal64 __rhs)

- decimal32 & **operator/=** (int __rhs)
- decimal32 & **operator/=** (long __rhs)
- decimal32 & **operator/=** (long long __rhs)
- decimal32 & **operator/=** (unsigned int __rhs)
- decimal32 & **operator/=** (unsigned long __rhs)
- decimal32 & **operator/=** (unsigned long long __rhs)

6.381.1 Detailed Description

3.2.2 Class decimal32.

6.381.2 Constructor & Destructor Documentation

decimal32()

```
std::decimal::decimal32::decimal32 (
            __decfloat32 __z)  [inline]
```

Conforming extension: Conversion from scalar decimal type.
The documentation for this class was generated from the following file:

- decimal

6.382 std::decimal::decimal64 Class Reference

```
#include <decimal>
```

Public Types

- typedef float **__decfloat64**

Public Member Functions

- decimal64 (__decfloat64 __z)
- **decimal64** (decimal128 d128)
- **decimal64** (decimal32 d32)
- **decimal64** (double __r)
- **decimal64** (float __r)
- **decimal64** (int __z)
- **decimal64** (long __z)
- **decimal64** (long double __r)
- **decimal64** (long long __z)
- **decimal64** (unsigned int __z)
- **decimal64** (unsigned long __z)
- **decimal64** (unsigned long long __z)
- __decfloat64 **__getval** (void)
- void **__setval** (__decfloat64 __x)
- **operator long long** () const
- decimal64 & **operator*=** (decimal128 __rhs)
- decimal64 & **operator*=** (decimal32 __rhs)
- decimal64 & **operator*=** (decimal64 __rhs)
- decimal64 & **operator*=** (int __rhs)
- decimal64 & **operator*=** (long __rhs)
- decimal64 & **operator*=** (long long __rhs)
- decimal64 & **operator*=** (unsigned int __rhs)

- decimal64 & **operator*=** (unsigned long __rhs)
- decimal64 & **operator*=** (unsigned long long __rhs)
- decimal64 & **operator++** ()
- decimal64 **operator++** (int)
- decimal64 & **operator+=** (decimal128 __rhs)
- decimal64 & **operator+=** (decimal32 __rhs)
- decimal64 & **operator+=** (decimal64 __rhs)
- decimal64 & **operator+=** (int __rhs)
- decimal64 & **operator+=** (long __rhs)
- decimal64 & **operator+=** (long long __rhs)
- decimal64 & **operator+=** (unsigned int __rhs)
- decimal64 & **operator+=** (unsigned long __rhs)
- decimal64 & **operator+=** (unsigned long long __rhs)
- decimal64 & **operator--** ()
- decimal64 **operator--** (int)
- decimal64 & **operator-=** (decimal128 __rhs)
- decimal64 & **operator-=** (decimal32 __rhs)
- decimal64 & **operator-=** (decimal64 __rhs)
- decimal64 & **operator-=** (int __rhs)
- decimal64 & **operator-=** (long __rhs)
- decimal64 & **operator-=** (long long __rhs)
- decimal64 & **operator-=** (unsigned int __rhs)
- decimal64 & **operator-=** (unsigned long __rhs)
- decimal64 & **operator-=** (unsigned long long __rhs)
- decimal64 & **operator/=** (decimal128 __rhs)
- decimal64 & **operator/=** (decimal32 __rhs)
- decimal64 & **operator/=** (decimal64 __rhs)
- decimal64 & **operator/=** (int __rhs)
- decimal64 & **operator/=** (long __rhs)
- decimal64 & **operator/=** (long long __rhs)
- decimal64 & **operator/=** (unsigned int __rhs)
- decimal64 & **operator/=** (unsigned long __rhs)
- decimal64 & **operator/=** (unsigned long long __rhs)

6.382.1 Detailed Description

3.2.3 Class decimal64.

6.382.2 Constructor & Destructor Documentation

decimal64()

```
std::decimal::decimal64::decimal64 (
            __decfloat64 __z)  [inline]
```

Conforming extension: Conversion from scalar decimal type.
The documentation for this class was generated from the following file:

- decimal

6.383 parallelism_v2::simd_abi::deduce< _Tp, _Np,... > Struct Template Reference

```
#include <simd.h>
```

6.383.1 Detailed Description

template<typename _Tp, size_t _Np, typename...>
struct parallelism_v2::simd_abi::deduce< _Tp, _Np,... >

Template Parameters

_Tp	The requested `value_type` for the elements.
_Np	The requested number of elements.
_Abis	This parameter is ignored, since this implementation cannot make any use of it. Either __a good native ABI is matched and used as `type` alias, or the `fixed_size<_Np>` ABI is used, which internally is built from the best matching native ABIs.

The documentation for this struct was generated from the following file:

- simd.h

6.384 __gnu_pbds::detail::default_comb_hash_fn Struct Reference

`#include <standard_policies.hpp>`

Public Types

- typedef direct_mask_range_hashing type

6.384.1 Detailed Description

Primary template, default_comb_hash_fn.

6.384.2 Member Typedef Documentation

type

`typedef direct_mask_range_hashing __gnu_pbds::detail::default_comb_hash_fn::type`
Dispatched type.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

6.385 std::default_delete< _Tp > Struct Template Reference

`#include <memory>`

Public Member Functions

- constexpr default_delete () noexcept=default
- template<typename _Up, typename = _Require<is_convertible<_Up*, _Tp*>>>
  constexpr default_delete (const default_delete< _Up > &) noexcept
- constexpr void operator() (_Tp *__ptr) const

6.385.1 Detailed Description

template<typename _Tp>
struct std::default_delete< _Tp >

Primary template of default_delete, used by unique_ptr for single objects

Since

C++11

6.385.2 Constructor & Destructor Documentation

default_delete() [1/2]

```
template<typename _Tp>
std::default_delete< _Tp >::default_delete ()  [constexpr], [default], [noexcept]
```

Default constructor.
Referenced by default_delete().

default_delete() [2/2]

```
template<typename _Tp>
template<typename _Up, typename = _Require<is_convertible<_Up*, _Tp*>>>
std::default_delete< _Tp >::default_delete (
            const default_delete< _Up > & )  [inline], [constexpr], [noexcept]
```

Converting constructor.
Allows conversion from a deleter for objects of another type, _Up, only if _Up* is convertible to _Tp*.
References default_delete().

6.385.3 Member Function Documentation

operator()()

```
template<typename _Tp>
void std::default_delete< _Tp >::operator() (
            _Tp * __ptr) const  [inline], [constexpr]
```

Calls `delete __ptr`.
The documentation for this struct was generated from the following file:

- unique_ptr.h

6.386 std::default_delete< _Tp[]> Struct Template Reference

`#include <memory>`

Public Member Functions

- constexpr default_delete () noexcept=default
- template<typename _Up, typename = _Require<is_convertible<_Up(*)[], _Tp(*)[]>>>
  constexpr default_delete (const default_delete< _Up[]> &) noexcept
- template<typename _Up>
  constexpr enable_if< is_convertible< _Up(*)[], _Tp(*)[]>::value >::type operator() (_Up *__ptr) const

6.386.1 Detailed Description

template<typename _Tp>
struct std::default_delete< _Tp[]>

Specialization of default_delete for arrays, used by `unique_ptr<T[]>`

Since

C++11

6.386.2 Constructor & Destructor Documentation

default_delete() [1/2]

```
template<typename _Tp>
std::default_delete< _Tp[]>::default_delete ()  [constexpr], [default], [noexcept]
```

Default constructor.
Referenced by default_delete().

default_delete() [2/2]

```
template<typename _Tp>
template<typename _Up, typename = _Require<is_convertible<_Up(*)[], _Tp(*)[]>>>
std::default_delete< _Tp[]>::default_delete (
            const default_delete< _Up[]> & )  [inline], [constexpr], [noexcept]
```

Converting constructor.
Allows conversion from a deleter for arrays of another type, such as a const-qualified version of `_Tp`.
Conversions from types derived from `_Tp` are not allowed because it is undefined to `delete[]` an array of derived types through a pointer to the base type.
References default_delete().

6.386.3 Member Function Documentation

operator()()

```
template<typename _Tp>
template<typename _Up>
enable_if< is_convertible< _Up(*)[], _Tp(*)[]>::value >::type std::default_delete< _Tp[]>::↩
operator() (
            _Up * __ptr) const  [inline], [constexpr]
```

Calls `delete[] __ptr`.
The documentation for this struct was generated from the following file:

- unique_ptr.h

6.387 __gnu_pbds::detail::default_eq_fn< Key > Struct Template Reference

`#include <standard_policies.hpp>`

Public Types

- typedef std::equal_to< Key > type

6.387.1 Detailed Description

template<typename Key>
struct __gnu_pbds::detail::default_eq_fn< Key >

Primary template, default_eq_fn.

6.387.2 Member Typedef Documentation

type

```
template<typename Key>
typedef std::equal_to<Key> __gnu_pbds::detail::default_eq_fn< Key >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

6.388 __gnu_pbds::detail::default_hash_fn< Key > Struct Template Reference

`#include <standard_policies.hpp>`

Public Types

- typedef std::tr1::hash< Key > type

6.388.1 Detailed Description

template<typename Key>
struct __gnu_pbds::detail::default_hash_fn< Key >

Primary template, default_hash_fn.

6.388.2 Member Typedef Documentation

type

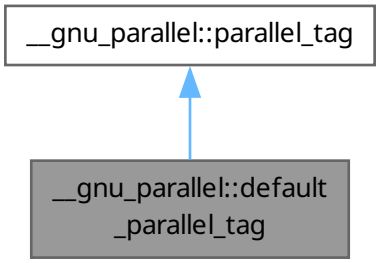
```
template<typename Key>
typedef std::tr1::hash<Key> __gnu_pbds::detail::default_hash_fn< Key >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

6.389 __gnu_parallel::default_parallel_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::default_parallel_tag:



Public Member Functions

- **default_parallel_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

6.389.1 Detailed Description

Recommends parallel execution using the default parallel algorithm.

6.389.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

6.390 __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn > Struct Template Reference

```
#include <standard_policies.hpp>
```

Public Types

- typedef cond_type::__type type

6.390.1 Detailed Description

template<typename Comb_Probe_Fn>
struct __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >

Primary template, default_probe_fn.

6.390.2 Member Typedef Documentation

type

```
template<typename Comb_Probe_Fn>
typedef cond_type::__type __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

6.391 __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn > Struct Template Reference

`#include <standard_policies.hpp>`

Public Types

- typedef hash_standard_resize_policy< size_policy_type, trigger, false, size_type > type

6.391.1 Detailed Description

template<typename Comb_Hash_Fn>
struct __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >

Primary template, default_resize_policy.

6.391.2 Member Typedef Documentation

type

```
template<typename Comb_Hash_Fn>
typedef hash_standard_resize_policy<size_policy_type, trigger, false, size_type> __gnu_pbds::detail::default_resi
Comb_Hash_Fn >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

6.392 std::default_sentinel_t Struct Reference

`#include <iterator_concepts.h>`

6.392.1 Detailed Description

A sentinel type that can be used to check for the end of a range.
For some iterator types the past-the-end sentinel value is independent of the underlying sequence, and a default sentinel can be used with them. For example, a `std::counted_iterator` keeps a count of how many elements remain, and so checking for the past-the-end value only requires checking if that count has reached zero. A past-the-end `std::istream_iterator` is equal to the default-constructed value, which can be easily checked.
Comparing iterators of these types to `std::default_sentinel` is a convenient way to check if the end has been reached.

Since

C++20

The documentation for this struct was generated from the following file:

- iterator_concepts.h

6.393 __gnu_pbds::detail::default_trie_access_traits< Key > Struct Template Reference

6.393.1 Detailed Description

template<typename Key>
struct __gnu_pbds::detail::default_trie_access_traits< Key >

Primary template, default_trie_access_traits.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

6.394 __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > > Struct Template Reference

`#include <standard_policies.hpp>`

Public Types

- typedef trie_string_access_traits< string_type > type

6.394.1 Detailed Description

template<typename Char, typename Char_Traits>
struct __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >

Partial specialization, default_trie_access_traits.

6.394.2 Member Typedef Documentation

type

```
template<typename Char, typename Char_Traits>
typedef trie_string_access_traits<string_type> __gnu_pbds::detail::default_trie_access_traits<
std::basic_string< Char, Char_Traits, std::allocator< char > > >::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

6.395 __gnu_pbds::detail::default_update_policy Struct Reference

`#include <standard_policies.hpp>`

Public Types

- typedef lu_move_to_front_policy type

6.395.1 Detailed Description

Default update policy.

6.395.2 Member Typedef Documentation

type

```
typedef lu_move_to_front_policy __gnu_pbds::detail::default_update_policy::type
```

Dispatched type.
The documentation for this struct was generated from the following file:

- standard_policies.hpp

6.396 std::defer_lock_t Struct Reference

`#include <std_mutex.h>`

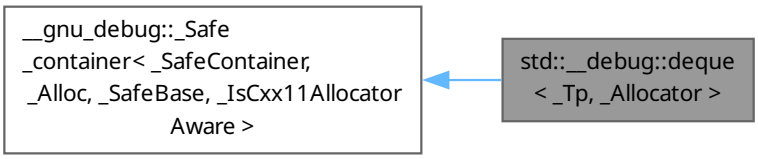
6.396.1 Detailed Description

Do not acquire ownership of the mutex.
The documentation for this struct was generated from the following file:

- std_mutex.h

6.397 std::__debug::deque< _Tp, _Allocator > Class Template Reference

`#include <deque>`
Inheritance diagram for std::__debug::deque< _Tp, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, deque > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, deque > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **deque** (_Base_ref __x)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  deque (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **deque** (const _Allocator &__a)
- **deque** (const deque &)=default
- **deque** (const deque &__d, const __type_identity_t< _Allocator > &__a)
- **deque** (deque &&)=default
- **deque** (deque &&__d, const __type_identity_t< _Allocator > &__a)
- **deque** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- **deque** (size_type __n, const __type_identity_t< _Tp > &__value, const _Allocator &__a=_Allocator())
- **deque** (size_type __n, const _Allocator &__a=_Allocator())
- const _Base & **_M_base** () const noexcept

- _Base & **_M_base** () noexcept
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (initializer_list< value_type > __l)
- void **assign** (size_type __n, const _Tp &__t)
- const_reference **back** () const noexcept
- reference **back** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (const_iterator __position, _Args &&... __args)
- template<typename... _Args>
  reference **emplace_back** (_Args &&... __args)
- template<typename... _Args>
  reference **emplace_front** (_Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __position)
- const_reference **front** () const noexcept
- reference **front** () noexcept
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator **insert** (const_iterator __position, _InputIterator __first, _InputIterator __last)
- iterator **insert** (const_iterator __position, _Tp &&__x)
- iterator **insert** (const_iterator __position, const _Tp &__x)
- iterator **insert** (const_iterator __position, initializer_list< value_type > __l)
- iterator **insert** (const_iterator __position, size_type __n, const _Tp &__x)
- deque & **operator=** (const deque &)=default
- deque & **operator=** (deque &&)=default
- deque & **operator=** (initializer_list< value_type > __l)
- const_reference **operator[ ]** (size_type __n) const noexcept
- reference **operator[ ]** (size_type __n) noexcept
- void **pop_back** () noexcept
- void **pop_front** () noexcept
- void **push_back** (_Tp &&__x)
- void **push_back** (const _Tp &__x)
- void **push_front** (_Tp &&__x)
- void **push_front** (const _Tp &__x)
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rend** () const noexcept
- reverse_iterator **rend** () noexcept
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const _Tp &__c)
- void **shrink_to_fit** () noexcept
- void **swap** (deque &__x) noexcept(/*conditional */)

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

6.397.1 Detailed Description

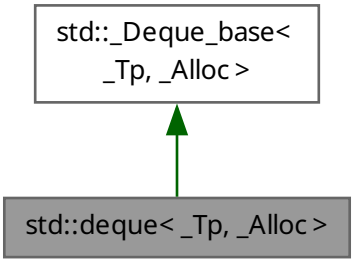
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__debug::deque< _Tp, _Allocator >

Class std::deque with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/deque

6.398 std::deque< _Tp, _Alloc > Class Template Reference

`#include <stl_deque.h>`
Inheritance diagram for std::deque< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef _Base::iterator **iterator**
- typedef _Alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- deque ()=default
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  deque (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- deque (const allocator_type &__a)
- deque (const deque &__x)
- deque (const deque &__x, const __type_identity_t< allocator_type > &__a)
- deque (deque &&)=default
- deque (deque &&__x, const __type_identity_t< allocator_type > &__a)
- deque (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- deque (size_type __n, const allocator_type &__a=allocator_type())
- deque (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- ~deque ()
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  void assign (_InputIterator __first, _InputIterator __last)
- void assign (initializer_list< value_type > __l)
- void assign (size_type __n, const value_type &__val)
- reference at (size_type __n)
- const_reference at (size_type __n) const
- const_reference back () const noexcept
- reference back () noexcept
- const_iterator begin () const noexcept
- iterator begin () noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- template<typename... _Args>
  iterator emplace (const_iterator __position, _Args &&... __args)
- template<typename... _Args>
  reference **emplace_back** (_Args &&... __args)
- template<typename... _Args>
  reference **emplace_front** (_Args &&... __args)
- bool empty () const noexcept
- const_iterator end () const noexcept
- iterator end () noexcept
- iterator erase (const_iterator __first, const_iterator __last)
- iterator erase (const_iterator __position)
- const_reference front () const noexcept
- reference front () noexcept
- allocator_type get_allocator () const noexcept
- iterator insert (const_iterator __p, initializer_list< value_type > __l)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)
- iterator insert (const_iterator __position, const value_type &__x)
- iterator insert (const_iterator __position, size_type __n, const value_type &__x)
- iterator insert (const_iterator __position, value_type &&__x)
- size_type max_size () const noexcept
- deque & operator= (const deque &__x)
- deque & operator= (deque &&__x) noexcept(_Alloc_traits::_S_always_equal())

- deque & operator= (initializer_list< value_type > __l)
- const_reference operator[ ] (size_type __n) const noexcept
- reference operator[ ] (size_type __n) noexcept
- void pop_back () noexcept
- void pop_front () noexcept
- void push_back (const value_type &__x)
- void **push_back** (value_type &&__x)
- void push_front (const value_type &__x)
- void **push_front** (value_type &&__x)
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rend () const noexcept
- reverse_iterator rend () noexcept
- void resize (size_type __new_size)
- void resize (size_type __new_size, const value_type &__x)
- void shrink_to_fit () noexcept
- size_type size () const noexcept
- void swap (deque &__x) noexcept

Protected Types

- enum { **_S_initial_map_size** }
- typedef __gnu_cxx::__alloc_traits< _Map_alloc_type > **_Map_alloc_traits**
- typedef _Alloc_traits::template rebind< _Ptr >::other **_Map_alloc_type**
- typedef _Alloc_traits::pointer **_Ptr**
- typedef _Alloc_traits::const_pointer **_Ptr_const**

Protected Member Functions

- template<typename _ForwardIterator>
  void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  void **_M_assign_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- void **_M_deallocate_map** (_Map_pointer __p, size_t __n) noexcept
- void **_M_default_append** (size_type __n)
- void **_M_default_initialize** ()
- template<typename _Alloc1>
  void **_M_destroy_data** (iterator __first, iterator __last, const _Alloc1 &)
- void **_M_destroy_data** (iterator __first, iterator __last, const std::allocator< _Tp > &)
- void **_M_destroy_data_aux** (iterator __first, iterator __last)
- template<typename... _Args>
  iterator **_M_emplace_aux** (iterator __pos, _Args &&... __args)
- iterator **_M_erase** (iterator __first, iterator __last)
- iterator **_M_erase** (iterator __pos)
- void **_M_erase_at_begin** (iterator __pos)
- void **_M_erase_at_end** (iterator __pos)
- void **_M_fill_assign** (size_type __n, const value_type &__val)
- void _M_fill_initialize (const value_type &__value)
- void **_M_fill_insert** (iterator __pos, size_type __n, const value_type &__x)
- _Map_alloc_type **_M_get_map_allocator** () const noexcept
- template<typename _ForwardIterator>
  void **_M_insert_aux** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, size_type __n)

- iterator **_M_insert_aux** (iterator __pos, const value_type &__x)
- void **_M_insert_aux** (iterator __pos, size_type __n, const value_type &__x)
- void **_M_move_assign1** (deque &&__x, false_type)
- void **_M_move_assign1** (deque &&__x, true_type) noexcept
- void **_M_move_assign2** (deque &&__x, false_type)
- void **_M_move_assign2** (deque &&__x, true_type)
- template<typename _InputIterator, typename _Sentinel>
  void **_M_range_append** (_InputIterator __first, _Sentinel __last, size_type __n)
- void _M_range_check (size_type __n) const
- template<typename _ForwardIterator>
  void **_M_range_insert_aux** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  void **_M_range_insert_aux** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _InputIterator, typename _Sentinel>
  void **_M_range_prepend** (_InputIterator __first, _Sentinel __last, size_type __n)
- template<typename... _Args>
  void **_M_replace_map** (_Args &&... __args)
- bool **_M_shrink_to_fit** ()

- template<typename _InputIterator>
  void _M_range_initialize (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator>
  void _M_range_initialize (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)

- template<typename... _Args>
  void _M_push_back_aux (_Args &&... __args)
- template<typename... _Args>
  void _M_push_front_aux (_Args &&... __args)
- void _M_pop_back_aux ()
- void _M_pop_front_aux ()

- iterator _M_reserve_elements_at_front (size_type __n)
- iterator _M_reserve_elements_at_back (size_type __n)
- void _M_new_elements_at_front (size_type __new_elements)
- void _M_new_elements_at_back (size_type __new_elements)

- void _M_reserve_map_at_back (size_type __nodes_to_add=1)
- void _M_reserve_map_at_front (size_type __nodes_to_add=1)
- void _M_reallocate_map (size_type __nodes_to_add, bool __add_at_front)

Static Protected Member Functions

- static size_t **_S_check_init_len** (size_t __n, const allocator_type &__a)
- static size_type **_S_max_size** (const _Tp_alloc_type &__a) noexcept

6.398.1 Detailed Description

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::deque< _Tp, _Alloc >

A standard container using fixed-size memory allocation and constant-time manipulation of elements at either end.

Template Parameters

_Tp	Type of element.
_Alloc	Allocator type, defaults to allocator<_Tp>.

Meets the requirements of a container, a reversible container, and a sequence, including the optional sequence requirements.
In previous HP/SGI versions of deque, there was an extra template parameter so users could control the node size. This extension turned out to violate the C++ standard (it can be detected using template template parameters), and it was removed.
Here's how a deque<Tp> manages memory. Each deque has 4 members:

- Tp** _M_map
- size_t _M_map_size
- iterator _M_start, _M_finish

map_size is at least 8. map is an array of map_size pointers-to-*nodes*. (The name map has nothing to do with the std::map class, and **nodes** should not be confused with std::list's usage of *node*.)
A *node* has no specific type name as such, but it is referred to as *node* in this file. It is a simple array-of-Tp. If Tp is very large, there will be one Tp element per node (i.e., an *array* of one). For non-huge Tp's, node size is inversely related to Tp size: the larger the Tp, the fewer Tp's will fit in a node. The goal here is to keep the total size of a node relatively small and constant over different Tp's, to improve allocator efficiency.
Not every pointer in the map array will point to a node. If the initial number of elements in the deque is small, the /middle/ map pointers will be valid, and the ones at the edges will be unused. This same situation will arise as the map grows: available map pointers, if any, will be on the ends. As new nodes are created, only a subset of the map's pointers need to be copied *outward*.
Class invariants:

- For any nonsingular iterator i:
  - i.node points to a member of the map array. (Yes, you read that correctly: i.node does not actually point to a node.) The member of the map array is what actually points to the node.
  - i.first == *(i.node) (This points to the node (first Tp element).)
  - i.last == i.first + node_size
  - i.cur is a pointer in the range [i.first, i.last). NOTE: the implication of this is that i.cur is always a dereferenceable pointer, even if i is a past-the-end iterator.
- Start and Finish are always nonsingular iterators. NOTE: this means that an empty deque must have one node, a deque with <N elements (where N is the node buffer size) must have one node, a deque with N through (2N-1) elements must have two nodes, etc.
- For every node other than start.node and finish.node, every element in the node is an initialized object. If start.↩ node == finish.node, then [start.cur, finish.cur) are initialized objects, and the elements outside that range are uninitialized storage. Otherwise, [start.cur, start.last) and [finish.first, finish.cur) are initialized objects, and ↩ [start.first, start.cur) and [finish.cur, finish.last) are uninitialized storage.
- [map, map + map_size) is a valid, non-empty range.
- [start.node, finish.node] is a valid range contained within [map, map + map_size).

- A pointer in the range [map, map + map_size) points to an allocated node if and only if the pointer is in the range [start.node, finish.node].

  Here's the magic: nothing in deque is **aware** of the discontiguous storage!

  The memory setup and layout occurs in the parent, _Base, and the iterator class is entirely responsible for *leaping* from one node to the next. All the implementation routines for deque itself work only through the start and finish iterators. This keeps the routines simple and sane, and we can use other standard algorithms as well.

6.398.2 Constructor & Destructor Documentation

deque() [1/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque ()  [default]
```

Creates a deque with no elements.
Referenced by insert(), and operator=().

deque() [2/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            const allocator_type & __a)  [inline], [explicit]
```

Creates a deque with no elements.

Parameters

_↩ _a	An allocator object.

deque() [3/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            size_type __n,
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Creates a deque with default constructed elements.

Parameters

_↩ _n	The number of elements to initially create.
_↩ _a	An allocator.

This constructor fills the deque with *n* default constructed elements.

deque() [4/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            size_type __n,
            const value_type & __value,
            const allocator_type & __a = allocator_type())  [inline]
```

Creates a deque with copies of an exemplar element.

Parameters

__n	The number of elements to initially create.
__value	An element to copy.
__a	An allocator.

This constructor fills the deque with *__n* copies of *__value*.

deque() [5/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            const deque< _Tp, _Alloc > & __x)  [inline]
```

Deque copy constructor.

Parameters

_↩ _x	A deque of identical element and allocator types.

The newly-created deque uses a copy of the allocator object used by *__x* (unless the allocator traits dictate a different object).

deque() [6/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            deque< _Tp, _Alloc > && )  [default]
```

Deque move constructor.
The newly-created deque contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified deque.

deque() [7/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            const deque< _Tp, _Alloc > & __x,
            const __type_identity_t< allocator_type > & __a)  [inline]
```

Copy constructor with alternative allocator.

deque() [8/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            deque< _Tp, _Alloc > && __x,
            const __type_identity_t< allocator_type > & __a)  [inline]
```

Move constructor with alternative allocator.

deque() [9/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::deque (
            initializer_list< value_type > __l,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a deque from an initializer list.

Parameters

_↩ _*l*	An initializer_list.
_↩ _*a*	An allocator object.

Create a deque consisting of copies of the elements in the initializer_list __*l*.
This will call the element type's copy constructor N times (where N is __l.size()) and do no memory reallocation.

deque() [10/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::deque< _Tp, _Alloc >::deque (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a deque from a range.

Parameters

__*first*	An input iterator.
__*last*	An input iterator.
__*a*	An allocator object.

Create a deque consisting of copies of the elements from [__first, __last).
If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is distance(__first,__last)) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

~deque()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::deque< _Tp, _Alloc >::~deque ()  [inline]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

6.398.3 Member Function Documentation

_M_fill_initialize()

```
template<typename _Tp, typename _Alloc>
void deque::_M_fill_initialize (
            const value_type & __value)  [protected]
```

Fills the deque with copies of value.

Parameters

__value	Initial value.

Precondition

_M_start and _M_finish have already been initialized, but none of the deque's elements have yet been constructed.

This function is called only when the user provides an explicit size (with or without an explicit exemplar value).
References std::_Destroy().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::deque().

_M_new_elements_at_back()

```
template<typename _Tp, typename _Alloc>
void deque::_M_new_elements_at_back (
            size_type __new_elements)  [protected]
```

Memory-handling helpers for the previous internal insert functions.
References _M_reserve_map_at_back(), max_size(), and std::size().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::_M_reserve_elements_at_back().

_M_new_elements_at_front()

```
template<typename _Tp, typename _Alloc>
void deque::_M_new_elements_at_front (
            size_type __new_elements)  [protected]
```

Memory-handling helpers for the previous internal insert functions.
References _M_reserve_map_at_front(), max_size(), and std::size().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::_M_reserve_elements_at_front().

_M_pop_back_aux()

```
template<typename _Tp, typename _Alloc>
void deque::_M_pop_back_aux ()  [protected]
```

Helper functions for push_* and pop_*.
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::pop_back().

_M_pop_front_aux()

```
template<typename _Tp, typename _Alloc>
void deque::_M_pop_front_aux ()  [protected]
```

Helper functions for push_* and pop_*.
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::pop_front().

_M_push_back_aux()

```
template<typename _Tp, typename _Alloc>
template<typename... _Args>
void deque::_M_push_back_aux (
            _Args &&... __args)  [protected]
```

Helper functions for push_* and pop_*.
References _M_reserve_map_at_back(), std::forward(), max_size(), and std::size().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::push_back().

_M_push_front_aux()

```
template<typename _Tp, typename _Alloc>
template<typename... _Args>
void deque::_M_push_front_aux (
            _Args &&... __args)  [protected]
```

Helper functions for push_* and pop_*.
References _M_reserve_map_at_front(), std::forward(), max_size(), and std::size().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::push_front().

_M_range_check()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_range_check (
            size_type __n) const  [inline], [protected]
```

Safety check used only from at().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::at(), and std::deque< _Tp, polymorphic_allocator< _Tp > >::at().

_M_range_initialize() [1/2]

```
template<typename _Tp, typename _Alloc>
template<typename _ForwardIterator>
void deque::_M_range_initialize (
            _ForwardIterator __first,
            _ForwardIterator __last,
            std::forward_iterator_tag )  [protected]
```

Fills the deque with whatever is in [first,last).

Parameters

__first	An input iterator.
__last	An input iterator.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using push_back on each value from the iterator.
References std::_Destroy(), std::advance(), and std::distance().

_M_range_initialize() [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _InputIterator>
void deque::_M_range_initialize (
            _InputIterator __first,
```

```
            _InputIterator __last,
            std::input_iterator_tag )  [protected]
```

Fills the deque with whatever is in [first,last).

Parameters

__first	An input iterator.
__last	An input iterator.

If the iterators are actually forward iterators (or better), then the memory layout can be done all at once. Else we move forward using push_back on each value from the iterator.
References clear(), and push_back().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::deque(), and std::deque< _Tp, polymorphic_allocator< _Tp > >::de

_M_reallocate_map()

```
template<typename _Tp, typename _Alloc>
void deque::_M_reallocate_map (
            size_type __nodes_to_add,
            bool __add_at_front)  [protected]
```

Memory-handling helpers for the major map.
Makes sure the _M_map has space for new nodes. Does not actually add the nodes. Can invalidate _M_map pointers. (And consequently, deque iterators.)
References std::max(), and max_size().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::_M_reserve_map_at_back(), and std::deque< _Tp, polymorphic_allo

_M_reserve_elements_at_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_back (
            size_type __n)  [inline], [protected]
```

Memory-handling helpers for the previous internal insert functions.

_M_reserve_elements_at_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::_M_reserve_elements_at_front (
            size_type __n)  [inline], [protected]
```

Memory-handling helpers for the previous internal insert functions.

_M_reserve_map_at_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_back (
            size_type __nodes_to_add = 1)  [inline], [protected]
```

Memory-handling helpers for the major map.
Makes sure the _M_map has space for new nodes. Does not actually add the nodes. Can invalidate _M_map pointers. (And consequently, deque iterators.)
Referenced by _M_new_elements_at_back(), and _M_push_back_aux().

_M_reserve_map_at_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::_M_reserve_map_at_front (
```

```
            size_type __nodes_to_add = 1)  [inline], [protected]
```

Memory-handling helpers for the major map.
Makes sure the _M_map has space for new nodes. Does not actually add the nodes. Can invalidate _M_map pointers. (And consequently, deque iterators.)
Referenced by _M_new_elements_at_front(), and _M_push_front_aux().

assign() [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::deque< _Tp, _Alloc >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Assigns a range to a deque.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a deque with copies of the elements in the range [__first,__last).
Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

assign() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::assign (
            initializer_list< value_type > __l)  [inline]
```

Assigns an initializer list to a deque.

Parameters

__l	An initializer_list.

This function fills a deque with copies of the elements in the initializer_list *__l*.
Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

assign() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::assign (
            size_type __n,
            const value_type & __val)  [inline]
```

Assigns a given value to a deque.

Parameters

__n	Number of elements to be assigned.
__val	Value to be assigned.

This function fills a deque with *n* copies of the given value. Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

at() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::at (
            size_type __n)  [inline]
```

Provides access to the data contained in the deque.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws out_of_range if the check fails.

at() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::at (
            size_type __n) const  [inline]
```

Provides access to the data contained in the deque.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the deque. The function throws out_of_range if the check fails.

back() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::back () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the deque.

back() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::back ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the last element of the deque.

begin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::begin () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.

begin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::begin ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points to the first element in the deque. Iteration is done in ordinary element order.
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::deque(), std::deque< _Tp, polymorphic_allocator< _Tp > >::deque std::deque< _Tp, polymorphic_allocator< _Tp > >::~deque(), std::deque< _Tp, polymorphic_allocator< _Tp > >::clear(), std::deque< _Tp, polymorphic_allocator< _Tp > >::front(), std::deque< _Tp, polymorphic_allocator< _Tp > >::front(), std::deque< _Tp, polymorphic_allocator< _Tp > >::insert(), std::deque< _Tp, polymorphic_allocator< _Tp > >::insert(), std::deque< _Tp, polymorphic_allocator< _Tp > >::insert(), std::operator<=>(), and operator=().

cbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the deque. Iteration is done in ordinary element order.
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::insert(), std::deque< _Tp, polymorphic_allocator< _Tp > >::insert() and std::deque< _Tp, polymorphic_allocator< _Tp > >::insert().

cend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

clear()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
Referenced by _M_range_initialize().

crbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::crbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

crend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::crend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

emplace()

```
template<typename _Tp, typename _Alloc>
template<typename...  _Args>
deque< _Tp, _Alloc >::iterator deque::emplace (
            const_iterator __position,
            _Args &&...  __args)
```

Inserts an object in deque before specified iterator.

Parameters

__position	A const_iterator into the deque.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location.
References std::forward().
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::insert().

empty()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::deque< _Tp, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the deque is empty. (Thus begin() would equal end().)

end() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::deque< _Tp, _Alloc >::end () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

end() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::end ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points one past the last element in the deque. Iteration is done in ordinary element order.

Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::deque(), std::deque< _Tp, polymorphic_allocator< _Tp > >::deque std::deque< _Tp, polymorphic_allocator< _Tp > >::~deque(), std::deque< _Tp, polymorphic_allocator< _Tp > >::back(), std::deque< _Tp, polymorphic_allocator< _Tp > >::back(), std::operator<=>(), and operator=().

erase() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Remove a range of elements.

Parameters

__first	Iterator pointing to the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or end()).

This function will erase the elements in the range [__first,__last) and shorten the deque accordingly.

The user is cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Remove element at given position.

Parameters

__position	Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the deque by one.

The user is cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

front() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::front () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the deque.

front() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::front ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the first element of the deque.

get_allocator()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::deque< _Tp, _Alloc >::get_allocator () const  [inline], [nodiscard], [noexcept]
```

Get a copy of the memory allocation object.
Referenced by operator=().

insert() [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
            const_iterator __p,
            initializer_list< value_type > __l)  [inline]
```

Inserts an initializer list into the deque.

Parameters

_↩ _p	An iterator into the deque.
_↩ _l	An initializer_list.

Returns

An iterator that points to the inserted data.

This function will insert copies of the data in the initializer_list __l into the deque before the location specified by __p. This is known as *list insert*.

insert() [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::deque< _Tp, _Alloc >::insert (
            const_iterator __position,
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Inserts a range into the deque.

Parameters

__position	A const_iterator into the deque.
__first	An input iterator.
__last	An input iterator.

Returns

An iterator that points to the inserted data.

This function will insert copies of the data in the range [__first,__last) into the deque before the location specified by *__position*. This is known as *range insert*.

insert() [3/5]

```
template<typename _Tp, typename _Alloc>
deque< _Tp, _Alloc >::iterator deque::insert (
            const_iterator __position,
            const value_type & __x)
```

Inserts given value into deque before specified iterator.

Parameters

__position	A const_iterator into the deque.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location.
References deque(), insert(), push_back(), and push_front().
Referenced by insert().

insert() [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
            const_iterator __position,
            size_type __n,
            const value_type & __x)  [inline]
```

Inserts a number of copies of given data into the deque.

Parameters

__position	A const_iterator into the deque.
__n	Number of elements to be inserted.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by *__position*.

insert() [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::deque< _Tp, _Alloc >::insert (
            const_iterator __position,
            value_type && __x)  [inline]
```

Inserts given rvalue into deque before specified iterator.

Parameters

__position	A const_iterator into the deque.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location.

max_size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque< _Tp, _Alloc >::max_size () const  [inline], [nodiscard], [noexcept]
```

Returns the size() of the largest possible deque.
Referenced by _M_new_elements_at_back(), _M_new_elements_at_front(), _M_push_back_aux(), _M_push_front_aux(), and _M_reallocate_map().

operator=() [1/3]

```
template<typename _Tp, typename _Alloc>
deque< _Tp, _Alloc > & deque::operator= (
            const deque< _Tp, _Alloc > & __x)
```

Deque assignment operator.

Parameters

_↩ _x	A deque of identical element and allocator types.

All the elements of *x* are copied.
The newly-created deque uses a copy of the allocator object used by *__x* (unless the allocator traits dictate a different object).
References deque(), std::__addressof(), begin(), end(), get_allocator(), size(), and std::size().

operator=() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque & std::deque< _Tp, _Alloc >::operator= (
            deque< _Tp, _Alloc > && __x)  [inline], [noexcept]
```

Deque move assignment operator.

Parameters

_↩ _x	A deque of identical element and allocator types.

The contents of __x are moved into this deque (without copying, if the allocators permit it). __x is a valid, but unspecified deque.

operator=() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
deque & std::deque< _Tp, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Assigns an initializer list to a deque.

Parameters

↩ _↩ ↩ _↩ l	An initializer_list.

This function fills a deque with copies of the elements in the initializer_list __l.
Note that the assignment completely changes the deque and that the resulting deque's size is the same as the number of elements assigned.

operator[]() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::deque< _Tp, _Alloc >::operator[] (
            size_type __n) const  [inline], [nodiscard], [noexcept]
```

Subscript access to the data contained in the deque.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

operator[]() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::deque< _Tp, _Alloc >::operator[] (
            size_type __n)  [inline], [nodiscard], [noexcept]
```

Subscript access to the data contained in the deque.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

pop_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_back ()  [inline], [noexcept]
```

Removes last element.
This is a typical stack operation. It shrinks the deque by one.
Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop_back() is called.

pop_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::pop_front ()  [inline], [noexcept]
```

Removes first element.
This is a typical stack operation. It shrinks the deque by one.
Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop_front() is called.

push_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_back (
            const value_type & __x)  [inline]
```

Add data to the end of the deque.

Parameters

_↩ _x	Data to be added.

This is a typical stack operation. The function creates an element at the end of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.
Referenced by _M_range_initialize(), and insert().

push_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::push_front (
            const value_type & __x)  [inline]
```

Add data to the front of the deque.

Parameters

_↩ _x	Data to be added.

This is a typical stack operation. The function creates an element at the front of the deque and assigns the given data to it. Due to the nature of a deque this operation can be done in constant time.
Referenced by insert().

rbegin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rbegin () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

rbegin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rbegin ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the deque. Iteration is done in reverse element order.

rend() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::deque< _Tp, _Alloc >::rend () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

rend() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::deque< _Tp, _Alloc >::rend ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the deque. Iteration is done in reverse element order.

resize() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::resize (
            size_type __new_size)  [inline]
```

Resizes the deque to the specified number of elements.

Parameters

__new_size	Number of elements the deque should contain.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise default constructed elements are appended.

resize() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::resize (
```

```
            size_type __new_size,
            const value_type & __x)  [inline]
```

Resizes the deque to the specified number of elements.

Parameters

__new_size	Number of elements the deque should contain.
__x	Data with which new elements should be populated.

This function will resize the deque to the specified number of elements. If the number is smaller than the deque's current size the deque is truncated, otherwise the deque is extended and new elements are populated with given data.

shrink_to_fit()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::shrink_to_fit ()  [inline], [noexcept]
```

A non-binding request to reduce memory use.

size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::deque< _Tp, _Alloc >::size () const  [inline], [nodiscard], [noexcept]
```

Returns the number of elements in the deque.
Referenced by std::deque< _Tp, polymorphic_allocator< _Tp > >::_M_range_check(), std::operator<=>(), operator=(), std::deque< _Tp, polymorphic_allocator< _Tp > >::resize(), and std::deque< _Tp, polymorphic_allocator< _Tp > >::resize().

swap()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::deque< _Tp, _Alloc >::swap (
            deque< _Tp, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another deque.

Parameters

_↩ _x	A deque of the same element and allocator types.

This exchanges the elements between two deques in constant time. (Four pointers, so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(d1,d2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.
The documentation for this class was generated from the following files:

- stl_deque.h
- deque.tcc

6.399 std::destroying_delete_t Struct Reference

```
#include <new>
```

6.399.1 Detailed Description

Tag type used to declare a class-specific operator delete that can invoke the destructor before deallocating the memory.

The documentation for this struct was generated from the following file:

- new

6.400 std::tr2::direct_bases< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- typedef __reflection_typelist< __direct_bases(_Tp)... > **type**

6.400.1 Detailed Description

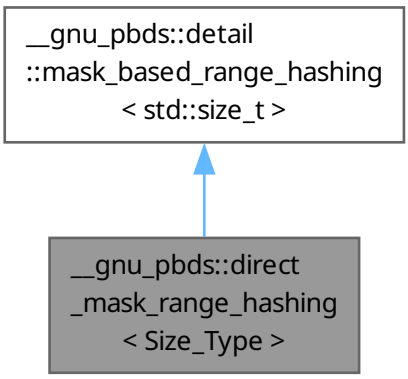
template<typename _Tp>
struct std::tr2::direct_bases< _Tp >

Enumerate all the direct base classes of a class. Form of a typelist.
The documentation for this struct was generated from the following file:

- tr2/type_traits

6.401 __gnu_pbds::direct_mask_range_hashing< Size_Type > Class Template Reference

`#include <hash_policy.hpp>`
Inheritance diagram for __gnu_pbds::direct_mask_range_hashing< Size_Type >:



Public Types

- typedef Size_Type **size_type**

Public Member Functions

- void **swap** (direct_mask_range_hashing< Size_Type > &other)

Protected Member Functions

- void **notify_resized** (size_type size)
- size_type operator() (size_type hash) const
- size_type **range_hash** (size_type hash) const
- void **swap** (mask_based_range_hashing &other)

6.401.1 Detailed Description

template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mask_range_hashing< Size_Type >

A mask range-hashing class (uses a bitmask).

6.401.2 Member Function Documentation

operator()()

```
template<typename Size_Type>
direct_mask_range_hashing< Size_Type >::size_type __gnu_pbds::direct_mask_range_hashing< Size_↩
Type >::operator() (
            size_type hash) const  [inline], [protected]
```

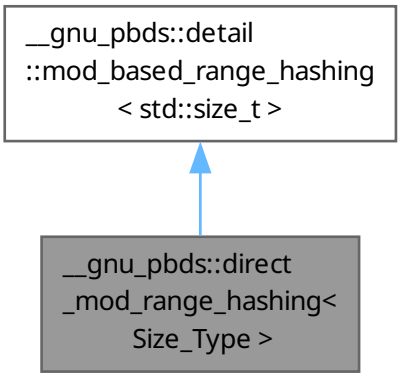
Transforms the __hash value hash into a ranged-hash value (using a bit-mask).
The documentation for this class was generated from the following file:

- hash_policy.hpp

6.402 __gnu_pbds::direct_mod_range_hashing< Size_Type > Class Template Reference

```
#include <hash_policy.hpp>
```

Inheritance diagram for __gnu_pbds::direct_mod_range_hashing< Size_Type >:



Public Types

- typedef Size_Type **size_type**

Public Member Functions

- void **swap** (direct_mod_range_hashing< Size_Type > &other)

Protected Member Functions

- void **notify_resized** (size_type s)
- void **notify_resized** (size_type size)
- size_type operator() (size_type hash) const
- size_type **range_hash** (size_type s) const
- void **swap** (mod_based_range_hashing &other)

6.402.1 Detailed Description

template<typename Size_Type = std::size_t>
class __gnu_pbds::direct_mod_range_hashing< Size_Type >

A mod range-hashing class (uses the modulo function).

6.402.2 Member Function Documentation

operator()()

```
template<typename Size_Type>
direct_mod_range_hashing< Size_Type >::size_type __gnu_pbds::direct_mod_range_hashing< Size_Type
>::operator() (
            size_type hash) const  [inline], [protected]
```

Transforms the __hash value hash into a ranged-hash value (using a modulo operation).
The documentation for this class was generated from the following file:

- hash_policy.hpp

6.403 std::filesystem::directory_entry Class Reference

```
#include <filesystem>
```

Public Member Functions

- **directory_entry** (const directory_entry &)=default
- **directory_entry** (const filesystem::path &__p)
- **directory_entry** (const filesystem::path &__p, error_code &__ec)
- **directory_entry** (directory_entry &&) noexcept=default
- void **assign** (const filesystem::path &__p)
- void **assign** (const filesystem::path &__p, error_code &__ec)
- bool **exists** () const
- bool **exists** (error_code &__ec) const noexcept
- uintmax_t **file_size** () const
- uintmax_t **file_size** (error_code &__ec) const noexcept
- uintmax_t **hard_link_count** () const
- uintmax_t **hard_link_count** (error_code &__ec) const noexcept
- bool **is_block_file** () const
- bool **is_block_file** (error_code &__ec) const noexcept
- bool **is_character_file** () const
- bool **is_character_file** (error_code &__ec) const noexcept
- bool **is_directory** () const

- bool **is_directory** (error_code &__ec) const noexcept
- bool **is_fifo** () const
- bool **is_fifo** (error_code &__ec) const noexcept
- bool **is_other** () const
- bool **is_other** (error_code &__ec) const noexcept
- bool **is_regular_file** () const
- bool **is_regular_file** (error_code &__ec) const noexcept
- bool **is_socket** () const
- bool **is_socket** (error_code &__ec) const noexcept
- bool **is_symlink** () const
- bool **is_symlink** (error_code &__ec) const noexcept
- file_time_type **last_write_time** () const
- file_time_type **last_write_time** (error_code &__ec) const noexcept
- **operator const filesystem::path &** () const noexcept
- strong_ordering **operator<=>** (const directory_entry &__rhs) const noexcept
- directory_entry & **operator=** (const directory_entry &)=default
- directory_entry & **operator=** (directory_entry &&) noexcept=default
- bool **operator==** (const directory_entry &__rhs) const noexcept
- const filesystem::path & **path** () const noexcept
- void **refresh** ()
- void **refresh** (error_code &__ec) noexcept
- void **replace_filename** (const filesystem::path &__p)
- void **replace_filename** (const filesystem::path &__p, error_code &__ec)
- file_status **status** () const
- file_status **status** (error_code &__ec) const noexcept
- file_status **symlink_status** () const
- file_status **symlink_status** (error_code &__ec) const noexcept

Friends

- struct **_Dir**
- class **directory_iterator**
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **operator<<** (basic_ostream< _CharT, _Traits > &__os, const directory_entry &__d)
- class **recursive_directory_iterator**

6.403.1 Detailed Description

The value type used by directory iterators.

Since

C++17

The documentation for this class was generated from the following file:

- bits/fs_dir.h

6.404 std::filesystem::directory_iterator Class Reference

`#include <filesystem>`

Public Types

- typedef ptrdiff_t **difference_type**
- typedef input_iterator_tag **iterator_category**
- typedef const directory_entry * **pointer**
- typedef const directory_entry & **reference**
- typedef directory_entry **value_type**

Public Member Functions

- **directory_iterator** (const directory_iterator &__rhs)=default
- **directory_iterator** (const path &__p)
- **directory_iterator** (const path &__p, directory_options __options)
- **directory_iterator** (const path &__p, directory_options __options, error_code &__ec)
- **directory_iterator** (const path &__p, error_code &__ec)
- **directory_iterator** (directory_iterator &&__rhs) noexcept=default
- directory_iterator & **increment** (error_code &__ec)
- const directory_entry & **operator*** () const noexcept
- directory_iterator & **operator++** ()
- __directory_iterator_proxy **operator++** (int)
- const directory_entry * **operator->** () const noexcept
- directory_iterator & **operator=** (const directory_iterator &__rhs)=default
- directory_iterator & **operator=** (directory_iterator &&__rhs) noexcept=default
- bool **operator==** (default_sentinel_t) const noexcept

Friends

- bool **operator==** (const directory_iterator &__lhs, const directory_iterator &__rhs) noexcept
- class **recursive_directory_iterator**

Related Symbols

(Note that these are not member symbols.)

- directory_iterator begin (directory_iterator __iter) noexcept
- directory_iterator end (directory_iterator) noexcept

- recursive_directory_iterator begin (recursive_directory_iterator __iter) noexcept
- recursive_directory_iterator end (recursive_directory_iterator) noexcept

6.404.1 Detailed Description

Iterator type for traversing the entries in a single directory.

Since

C++17

The documentation for this class was generated from the following file:

- bits/fs_dir.h

6.405 std::discard_block_engine< _RandomNumberEngine, __p, __r > Class Template Reference

`#include <random>`

Public Types

- template<typename _Sseq>
  using **_If_seed_seq**
- typedef _RandomNumberEngine::result_type result_type

Public Member Functions

- discard_block_engine ()
- discard_block_engine (_RandomNumberEngine &&__rng)
- template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
  discard_block_engine (_Sseq &__q)
- discard_block_engine (const _RandomNumberEngine &__rng)
- discard_block_engine (result_type __s)
- const _RandomNumberEngine & base () const noexcept
- void discard (unsigned long long __z)
- result_type operator() ()
- void seed ()
- template<typename _Sseq>
  _If_seed_seq< _Sseq > seed (_Sseq &__q)
- void seed (result_type __s)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr size_t **block_size**
- static constexpr size_t **used_block**

Friends

- template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > &__x)
- bool operator== (const discard_block_engine &__lhs, const discard_block_engine &__rhs)
- template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > &__x)

6.405.1 Detailed Description

template<typename _RandomNumberEngine, size_t __p, size_t __r>
class std::discard_block_engine< _RandomNumberEngine, __p, __r >

Produces random numbers from some base engine by discarding blocks of data.

Precondition

$0 \leq r \leq p$

Since

C++11

6.405.2 Member Typedef Documentation

result_type

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
typedef _RandomNumberEngine::result_type std::discard_block_engine< _RandomNumberEngine, __p, __r
>::result_type
```

The type of the generated random value.

6.405.3 Constructor & Destructor Documentation

discard_block_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine ()  [inline]
```

Constructs a default discard_block_engine engine.
The underlying engine is default constructed as well.

discard_block_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
            const _RandomNumberEngine & __rng)  [inline], [explicit]
```

Copy constructs a discard_block_engine engine.
Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

discard_block_engine() [3/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
            _RandomNumberEngine && __rng)  [inline], [explicit]
```

Move constructs a discard_block_engine engine.
Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

discard_block_engine() [4/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
            result_type __s)  [inline], [explicit]
```

Seed constructs a discard_block_engine engine.
Constructs the underlying generator engine seeded with __s.

Parameters

_↩ _s	A seed value for the base class engine.

discard_block_engine() [5/5]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard_block_engine (
            _Sseq & __q)  [inline], [explicit]
```

Generator construct a discard_block_engine engine.

Parameters

_↩ _q	A seed sequence.

6.405.4 Member Function Documentation

base()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
const _RandomNumberEngine & std::discard_block_engine< _RandomNumberEngine, __p, __r >::base ()
const  [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

discard()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r >::discard (
            unsigned long long __z)  [inline]
```

Discard a sequence of random numbers.

max()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
constexpr result_type std::discard_block_engine< _RandomNumberEngine, __p, __r >::max ()  [inline],
[static], [constexpr]
```

Gets the maximum value in the generated random number range.

min()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
constexpr result_type std::discard_block_engine< _RandomNumberEngine, __p, __r >::min ()  [inline],
```

[static], [constexpr]
Gets the minimum value in the generated random number range.

operator()()

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
discard_block_engine< _RandomNumberEngine, __p, __r >::result_type std::discard_block_engine< _↩
RandomNumberEngine, __p, __r >::operator() ()
```
Gets the next value in the generated random number sequence.

seed() [1/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r >::seed ()  [inline]
```
Reseeds the discard_block_engine object with the default seed for the underlying base class generator engine.

seed() [2/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::discard_block_engine< _RandomNumberEngine, __p, __r >::seed (
            _Sseq & __q)  [inline]
```
Reseeds the discard_block_engine object with the given seed sequence.

Parameters

_↩ _q	A seed generator function.

seed() [3/3]

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
void std::discard_block_engine< _RandomNumberEngine, __p, __r >::seed (
            result_type __s)  [inline]
```
Reseeds the discard_block_engine object with the default seed for the underlying base class generator engine.

6.405.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _↩
Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > & __x)  [friend]
```
Inserts the current state of a discard_block_engine random number generator engine __x into the output stream __os.

Parameters

__os	An output stream.
__x	A discard_block_engine random number generator engine.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
bool operator== (
             const discard_block_engine< _RandomNumberEngine, __p, __r > & __lhs,
             const discard_block_engine< _RandomNumberEngine, __p, __r > & __rhs)  [friend]
```

Compares two discard_block_engine random number generator objects of the same type for equality.

Parameters

__lhs	A discard_block_engine random number generator object.
__rhs	Another discard_block_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

operator>>

```
template<typename _RandomNumberEngine, size_t __p, size_t __r>
template<typename _RandomNumberEngine1, size_t __p1, size_t __r1, typename _CharT, typename _↩
Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
             std::basic_istream< _CharT, _Traits > & __is,
             std::discard_block_engine< _RandomNumberEngine1, __p1, __r1 > & __x)  [friend]
```

Extracts the current state of a % subtract_with_carry_engine random number generator engine __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A discard_block_engine random number generator engine.

Returns

The input stream with the state of __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.406 std::discrete_distribution< _IntType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- template<typename _InputIterator>
  discrete_distribution (_InputIterator __wbegin, _InputIterator __wend)
- **discrete_distribution** (const param_type &__p)
- **discrete_distribution** (initializer_list< double > __wl)
- template<typename _Func>
  discrete_distribution (size_t __nw, double __xmin, double __xmax, _Func __fw)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- std::vector< double > probabilities () const
- void reset ()

Friends

- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::discrete_distribution< _IntType1 > &__x)
- bool operator== (const discrete_distribution &__d1, const discrete_distribution &__d2)
- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::discrete_distribution< _IntType1 > &__x)

6.406.1 Detailed Description

template<typename _IntType = int>
class std::discrete_distribution< _IntType >

A discrete_distribution random number distribution.
This distribution produces random numbers $i, 0 \leq i < n$, distributed according to the probability mass function $p(i|p_0, ..., p_{n-1}) = p_i$.

Since

C++11

6.406.2 Member Typedef Documentation

result_type

```
template<typename _IntType = int>
typedef _IntType std::discrete_distribution< _IntType >::result_type
```

The type of the range of the distribution.

6.406.3 Member Function Documentation

max()

```
template<typename _IntType = int>
result_type std::discrete_distribution< _IntType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _IntType = int>
result_type std::discrete_distribution< _IntType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::discrete_distribution< _IntType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _IntType = int>
param_type std::discrete_distribution< _IntType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _IntType = int>
void std::discrete_distribution< _IntType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

probabilities()

```
template<typename _IntType = int>
std::vector< double > std::discrete_distribution< _IntType >::probabilities () const  [inline]
```

Returns the probabilities of the distribution.

reset()

```
template<typename _IntType = int>
void std::discrete_distribution< _IntType >::reset ()  [inline]
```

Resets the distribution state.

6.406.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::discrete_distribution< _IntType1 > & __x)  [friend]
```

Inserts a discrete_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A discrete_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _IntType = int>
bool operator== (
            const discrete_distribution< _IntType > & __d1,
            const discrete_distribution< _IntType > & __d2)  [friend]
```

Return true if two discrete distributions have the same parameters.

operator>>

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::discrete_distribution< _IntType1 > & __x)  [friend]
```

Extracts a discrete_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A discrete_distribution random number generator engine.

Returns

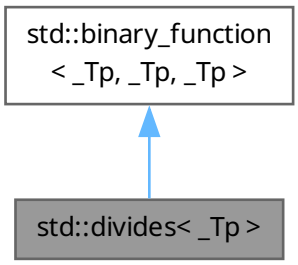
The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.407 std::divides< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::divides< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

6.407.1 Detailed Description

template<typename _Tp>
struct std::divides< _Tp >

One of the math functors.

6.407.2 Member Typedef Documentation

first_argument_type

typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type [inherited]
first_argument_type is the type of the first argument

result_type

typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type [inherited]
result_type is the return type

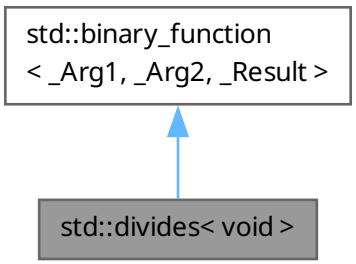
second_argument_type

typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.408 std::divides< void > Struct Reference

#include <stl_function.h>
Inheritance diagram for std::divides< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef void result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t)/↩ std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t)/std::forward< _Up >(__u))

6.408.1 Detailed Description

One of the math functors.

6.408.2 Member Typedef Documentation

first_argument_type

```
typedef void std::binary_function< void, void, void >::first_argument_type
```

`first_argument_type` is the type of the first argument

result_type

```
typedef void std::binary_function< void, void, void >::result_type
```

`result_type` is the return type

second_argument_type

```
typedef void std::binary_function< void, void, void >::second_argument_type
```

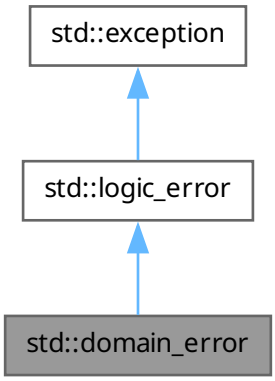
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.409 std::domain_error Class Reference

```
#include <stdexcept_except.h>
```

Inheritance diagram for std::domain_error:



Public Member Functions

- **domain_error** (const char *)
- **domain_error** (const domain_error &)=default
- **domain_error** (const string &__arg)
- **domain_error** (domain_error &&)=default

- domain_error & **operator=** (const domain_error &)=default
- domain_error & **operator=** (domain_error &&)=default
- virtual const char * what () const noexcept

6.409.1 Detailed Description

Thrown by the library, or by you, to report domain errors (domain in the mathematical sense).

6.409.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
References what().
Referenced by what().
The documentation for this class was generated from the following file:

- stdexcept_except.h

6.410 __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc > Struct Template Reference

```
#include <null_node_metadata.hpp>
```

Public Types

- typedef const_iterator **const_reference**
- typedef const_reference **reference**
- typedef const_iterator **value_type**

6.410.1 Detailed Description

template<typename Key, typename Data, typename _Alloc>
struct __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >

Constant node iterator.
The documentation for this struct was generated from the following file:

- null_node_metadata.hpp

6.411 std::chrono::duration< _Rep, _Period > Class Template Reference

Public Types

- using **period**
- using **rep**

Public Member Functions

- template<typename _Rep2, typename = _Require< is_convertible<const _Rep2&, rep>, __or_<__is_float<rep>, __not_<__is_float<↩ _Rep2>>>>>>
  constexpr **duration** (const _Rep2 &__rep)
- **duration** (const duration &)=default

- template<typename _Rep2, typename _Period2, typename = _Require< is_convertible<const _Rep2&, rep>, __or_<__is_float<rep>, __and_<__is_harmonic<_Period2>, __not_<__is_float<_Rep2>>>>>>
  constexpr **duration** (const duration< _Rep2, _Period2 > &__d)
- constexpr rep **count** () const
- template<typename _Rep2 = rep>
  constexpr __enable_if_t<!treat_as_floating_point< _Rep2 >::value, duration & > **operator%=** (const duration &__d)
- template<typename _Rep2 = rep>
  constexpr __enable_if_t<!treat_as_floating_point< _Rep2 >::value, duration & > **operator%=** (const rep &__rhs)
- constexpr duration & **operator*=** (const rep &__rhs)
- constexpr duration< typename common_type< rep >::type, period > **operator+** () const
- constexpr duration & **operator++** ()
- constexpr duration **operator++** (int)
- constexpr duration & **operator+=** (const duration &__d)
- constexpr duration< typename common_type< rep >::type, period > **operator-** () const
- constexpr duration & **operator--** ()
- constexpr duration **operator--** (int)
- constexpr duration & **operator-=** (const duration &__d)
- constexpr duration & **operator/=** (const rep &__rhs)
- duration & **operator=** (const duration &)=default

Static Public Member Functions

- static constexpr duration **max** () noexcept
- static constexpr duration **min** () noexcept
- static constexpr duration **zero** () noexcept

Related Symbols

(Note that these are not member symbols.)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type operator+ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, _Rep2 >, _Period > operator* (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool operator== (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

6.411.1 Detailed Description

template<typename _Rep, typename _Period>
class std::chrono::duration< _Rep, _Period >

`chrono::duration` represents a distance between two points in time
The documentation for this class was generated from the following file:

- chrono.h

6.412 std::chrono::duration_values< _Rep > Struct Template Reference

`#include <chrono.h>`

Static Public Member Functions

- static constexpr _Rep **max** () noexcept
- static constexpr _Rep **min** () noexcept
- static constexpr _Rep **zero** () noexcept

6.412.1 Detailed Description

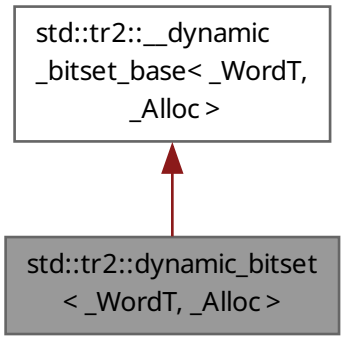
template<typename _Rep>
struct std::chrono::duration_values< _Rep >

duration_values
The documentation for this struct was generated from the following file:

- chrono.h

6.413 std::tr2::dynamic_bitset< _WordT, _Alloc > Class Template Reference

`#include <dynamic_bitset>`
Inheritance diagram for std::tr2::dynamic_bitset< _WordT, _Alloc >:



Classes

- class reference

Public Types

- typedef __dynamic_bitset_base< _WordT, _Alloc > **_Base**
- typedef _Alloc **allocator_type**
- typedef _WordT **block_type**
- typedef bool **const_reference**
- typedef size_t **size_type**

Public Member Functions

- dynamic_bitset ()=default
- dynamic_bitset (const allocator_type &__alloc)
- dynamic_bitset (const char *__str, const allocator_type &__alloc=allocator_type())
- dynamic_bitset (const dynamic_bitset &)=default
- template<typename _CharT, typename _Traits, typename _Alloc1>
  dynamic_bitset (const std::basic_string< _CharT, _Traits, _Alloc1 > &__str, typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __pos=0, typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __n=std::basic_string< _CharT, _Traits, _Alloc1 >::npos, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'), const allocator_type &__alloc=allocator_type())
- dynamic_bitset (dynamic_bitset &&__b) noexcept
- **dynamic_bitset** (initializer_list< block_type > __il, const allocator_type &__alloc=allocator_type())
- dynamic_bitset (size_type __nbits, unsigned long long __val=0ULL, const allocator_type &__alloc=allocator_type())
- template<typename _Traits = std::char_traits<char>, typename _CharT = typename _Traits::char_type>
  void **_M_copy_from_ptr** (const _CharT *, size_t, size_t, size_t, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'))
- template<typename _CharT, typename _Traits, typename _Alloc1>
  void **_M_copy_from_string** (const basic_string< _CharT, _Traits, _Alloc1 > &__str, size_t __pos, size_t __n, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1'))
- template<typename _CharT, typename _Traits, typename _Alloc1>
  void **_M_copy_to_string** (std::basic_string< _CharT, _Traits, _Alloc1 > &__str, _CharT __zero=_CharT('0'), _CharT __one=_CharT('1')) const
- bool all () const
- bool any () const
- template<typename _BlockInputIterator>
  void append (_BlockInputIterator __first, _BlockInputIterator __last)
- void append (block_type __block)
- void **append** (initializer_list< block_type > __il)
- void clear ()
- size_type count () const noexcept
- bool empty () const noexcept
- size_type find_first () const
- size_type find_next (size_t __prev) const
- dynamic_bitset & flip ()
- dynamic_bitset & flip (size_type __pos)
- allocator_type get_allocator () const noexcept
- bool **is_proper_subset_of** (const dynamic_bitset &__b) const
- bool **is_subset_of** (const dynamic_bitset &__b) const
- constexpr size_type max_size () noexcept
- bool none () const
- size_type num_blocks () const noexcept
- dynamic_bitset & operator= (const dynamic_bitset &)=default
- dynamic_bitset & operator= (dynamic_bitset &&__b) noexcept(std::is_nothrow_move_assignable< _Base >::value)
- dynamic_bitset operator~ () const
- void push_back (bool __bit)
- dynamic_bitset & reset ()
- dynamic_bitset & reset (size_type __pos)
- void resize (size_type __nbits, bool __value=false)
- dynamic_bitset & set ()

- dynamic_bitset & set (size_type __pos, bool __val=true)
- size_type size () const noexcept
- void swap (dynamic_bitset &__b) noexcept
- bool test (size_type __pos) const
- template<typename _CharT = char, typename _Traits = std::char_traits<_CharT>, typename _Alloc1 = std::allocator<_CharT>> std::basic_string< _CharT, _Traits, _Alloc1 > to_string (_CharT __zero=_CharT('0'), _CharT __one=_CharT('1')) const
- unsigned long long to_ullong () const
- unsigned long to_ulong () const

- dynamic_bitset & operator&= (const dynamic_bitset &__rhs)
- dynamic_bitset & operator&= (dynamic_bitset &&__rhs)
- dynamic_bitset & operator|= (const dynamic_bitset &__rhs)
- dynamic_bitset & operator^= (const dynamic_bitset &__rhs)
- dynamic_bitset & operator-= (const dynamic_bitset &__rhs)

- dynamic_bitset & operator<<= (size_type __pos)
- dynamic_bitset & operator>>= (size_type __pos)

- reference operator[] (size_type __pos)
- const_reference operator[] (size_type __pos) const

- dynamic_bitset operator<< (size_type __pos) const
- dynamic_bitset operator>> (size_type __pos) const

Static Public Attributes

- static const size_type **bits_per_block**
- static const size_type **npos**

Friends

- bool **operator<** (const dynamic_bitset &__lhs, const dynamic_bitset &__rhs) noexcept
- bool **operator==** (const dynamic_bitset &__lhs, const dynamic_bitset &__rhs) noexcept
- class **reference**

6.413.1 Detailed Description

template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
class std::tr2::dynamic_bitset< _WordT, _Alloc >

The dynamic_bitset class represents a sequence of bits.
See N2050, Proposal to Add a Dynamically Sizeable Bitset to the Standard Library. http://www.open-std.↩org/jtc1/sc22/wg21/docs/papers/2006/n2050.pdf
In the general unoptimized case, storage is allocated in word-sized blocks. Let B be the number of bits in a word, then (Nb+(B-1))/B words will be used for storage. B - NbB bits are unused. (They are the high-order bits in the highest word.) It is a class invariant that those unused bits are always zero.
If you think of dynamic_bitset as "a simple array of bits," be aware that your mental picture is reversed: a dynamic_bitset behaves the same way as bits in integers do, with the bit at index 0 in the "least significant / right-hand" position, and

the bit at index Nb-1 in the "most significant / left-hand" position. Thus, unlike other containers, a dynamic_bitset's index "counts from right to left," to put it very loosely.
This behavior is preserved when translating to and from strings. For example, the first line of the following program probably prints "b('a') is 0001100001" on a modern ASCII system.

```
#include <dynamic_bitset>
#include <iostream>
#include <sstream>

using namespace std;

int main()
{
    long         a = 'a';
    dynamic_bitset<> b(a);

    cout « "b('a') is " « b « endl;

    ostringstream s;
    s « b;
    string  str = s.str();
    cout « "index 3 in the string is " « str[3] « " but\n"
         « "index 3 in the bitset is " « b[3] « endl;
}
```

Most of the actual code isn't contained in dynamic_bitset<> itself, but in the base class __dynamic_bitset_base. The base class works with whole words, not with individual bits. This allows us to specialize __dynamic_bitset_base for the important special case where the dynamic_bitset is only a single word.
Extra confusion can result due to the fact that the storage for __dynamic_bitset_base *is* a vector, and is indexed as such. This is carefully encapsulated.

6.413.2 Constructor & Destructor Documentation

dynamic_bitset() [1/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset ()  [default]
```

All bits set to zero.
Referenced by dynamic_bitset(), dynamic_bitset(), operator&=(), operator&=(), operator-=(), operator<<(), operator<<=(), operator=(), operator=(), operator>>(), operator^=(), operator|=(), operator~(), and swap().

dynamic_bitset() [2/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
            const allocator_type & __alloc)  [inline], [explicit]
```

All bits set to zero.

dynamic_bitset() [3/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
            size_type __nbits,
            unsigned long long __val = 0ULL,
            const allocator_type & __alloc = allocator_type())  [inline], [explicit]
```

Initial bits bitwise-copied from a single word (others set to zero).

dynamic_bitset() [4/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _CharT, typename _Traits, typename _Alloc1>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
            const std::basic_string< _CharT, _Traits, _Alloc1 > & __str,
```

```
            typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __pos = 0,
            typename basic_string< _CharT, _Traits, _Alloc1 >::size_type __n = std::basic_string<↩
_CharT, _Traits, _Alloc1>::npos,
            _CharT __zero = _CharT('0'),
            _CharT __one = _CharT('1'),
            const allocator_type & __alloc = allocator_type())  [inline], [explicit]
```

Use a subset of a string.

Parameters

__str	A string of '0' and '1' characters.
__pos	Index of the first character in __str to use.
__n	The number of characters to copy.
__zero	The character to use for unset bits.
__one	The character to use for set bits.
__alloc	An allocator.

Exceptions

std::out_of_range	If __pos is bigger the size of __str.
std::invalid_argument	If a character appears in the string which is neither '0' nor '1'.

References std::basic_string< _CharT, _Traits, _Alloc >::npos, and std::basic_string< _CharT, _Traits, _Alloc >::size().

dynamic_bitset() [5/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
            const char * __str,
            const allocator_type & __alloc = allocator_type())  [inline], [explicit]
```

Construct from a string.

Parameters

__str	A string of '0' and '1' characters.
__alloc	An allocator.

Exceptions

std::invalid_argument	If a character appears in the string which is neither '0' nor '1'.

dynamic_bitset() [6/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
            const dynamic_bitset< _WordT, _Alloc > & )  [default]
```

Copy constructor.
References dynamic_bitset().

dynamic_bitset() [7/7]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
std::tr2::dynamic_bitset< _WordT, _Alloc >::dynamic_bitset (
            dynamic_bitset< _WordT, _Alloc > && __b)  [inline], [noexcept]
```

Move constructor.
References dynamic_bitset(), and std::move().

6.413.3 Member Function Documentation

all()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::all () const  [inline]
```

Tests whether all the bits are on.

Returns

True if all the bits are set.

any()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::any () const  [inline]
```

Tests whether any of the bits are on.

Returns

True if at least one bit is set.

append() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _BlockInputIterator>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::append (
            _BlockInputIterator __first,
            _BlockInputIterator __last)  [inline]
```

Append an iterator range of blocks.
References append().

append() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::append (
            block_type __block)  [inline]
```

Append a block.
Referenced by append().

clear()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::clear ()  [inline]
```

Clear the bitset.

count()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::count () const  [inline], [noexcept]
```

Returns the number of bits which are set.

empty()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the dynamic_bitset is empty.

find_first()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::find_first () const  [inline]
```

Finds the index of the first "on" bit.

Returns

The index of the first bit set, or size() if not found.

See also

find_next

find_next()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::find_next (
            size_t __prev) const  [inline]
```

Finds the index of the next "on" bit after prev.

Returns

The index of the next bit set, or size() if not found.

Parameters

__prev	Where to start searching.

See also

find_first

flip() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::flip ()  [inline]
```

Toggles every bit to its opposite value.

Referenced by operator~().

flip() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::flip (
            size_type __pos)  [inline]
```

Toggles a given bit to its opposite value.

Parameters

__pos	The index of the bit.

Exceptions

std::out_of_range	If *__pos* is bigger the size of the set.

get_allocator()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
allocator_type std::tr2::dynamic_bitset< _WordT, _Alloc >::get_allocator () const  [inline],
[noexcept]
```

Return the allocator for the bitset.
Referenced by operator=().

max_size()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::max_size ()  [inline], [constexpr], [noexcept]
```

Returns the maximum size of a dynamic_bitset object having the same type as *this. The real answer is max() * bits_per_block but is likely to overflow.
References std::numeric_limits< _Tp >::max().

none()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::none () const  [inline]
```

Tests whether any of the bits are on.

Returns

True if none of the bits are set.

num_blocks()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::num_blocks () const  [inline], [noexcept]
```

Returns the total number of blocks.

operator&=() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator&= (
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.
References dynamic_bitset().

operator&=() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator&= (
            dynamic_bitset< _WordT, _Alloc > && __rhs)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.
References dynamic_bitset(), and std::move().

operator-=()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator-= (
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.
References dynamic_bitset().

operator<<()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator<< (
            size_type __pos) const  [inline]
```

Self-explanatory.
References dynamic_bitset().

operator<<=()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator<<= (
            size_type __pos)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__pos	The number of places to shift.

These should be self-explanatory.
References dynamic_bitset().

operator=() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (
            const dynamic_bitset< _WordT, _Alloc > & )  [default]
```

Copy assignment operator.
References dynamic_bitset().

operator=() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator= (
            dynamic_bitset< _WordT, _Alloc > && __b)  [inline], [noexcept]
```

Move assignment operator.
References dynamic_bitset(), and get_allocator().

operator>>()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>> (
            size_type __pos) const  [inline]
```

Self-explanatory.
References dynamic_bitset().

operator>>=()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator>>= (
            size_type __pos)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__pos	The number of places to shift.

These should be self-explanatory.

operator[]() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
reference std::tr2::dynamic_bitset< _WordT, _Alloc >::operator[] (
            size_type __pos)  [inline]
```

Array-indexing support.

Parameters

__pos	Index into the dynamic_bitset.

Returns

A bool for a 'const dynamic_bitset'. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

operator[]() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
const_reference std::tr2::dynamic_bitset< _WordT, _Alloc >::operator[] (
            size_type __pos) const  [inline]
```

Array-indexing support.

Parameters

__pos	Index into the dynamic_bitset.

Returns

A bool for a 'const dynamic_bitset'. For non-const bitsets, an instance of the reference proxy class.

Note

These operators do no range checking and throw no exceptions, as required by DR 11 to the standard.

operator^=()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator^= (
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.
References dynamic_bitset().

operator"|=()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::operator|= (
            const dynamic_bitset< _WordT, _Alloc > & __rhs)  [inline]
```

Operations on dynamic_bitsets.

Parameters

__rhs	A same-sized dynamic_bitset.

These should be self-explanatory.
References dynamic_bitset().

operator~()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset std::tr2::dynamic_bitset< _WordT, _Alloc >::operator~ () const  [inline]
```

See the no-argument flip().
References dynamic_bitset(), and flip().

push_back()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::push_back (
            bool __bit)  [inline]
```

Push a bit onto the high end of the bitset.
References size().

reset() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::reset ()  [inline]
```

Sets every bit to false.

reset() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::reset (
            size_type __pos)  [inline]
```

Sets a given bit to false.

Parameters

__pos	The index of the bit.

Exceptions

std::out_of_range	If *__pos* is bigger the size of the set.

Same as writing `set(__pos, false)`.

resize()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::resize (
            size_type __nbits,
            bool __value = false)  [inline]
```

Resize the bitset.
Referenced by std::tr2::operator>>().

set() [1/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::set ()  [inline]
```

Sets every bit to true.

set() [2/2]

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
dynamic_bitset & std::tr2::dynamic_bitset< _WordT, _Alloc >::set (
            size_type __pos,
            bool __val = true)  [inline]
```

Sets a given bit to a particular value.

Parameters

__pos	The index of the bit.
__val	Either true or false, defaults to true.

Exceptions

std::out_of_range	If *__pos* is bigger the size of the set.

size()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
size_type std::tr2::dynamic_bitset< _WordT, _Alloc >::size () const  [inline], [noexcept]
```

Returns the total number of bits.

Referenced by std::tr2::operator>>(), and push_back().

swap()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
void std::tr2::dynamic_bitset< _WordT, _Alloc >::swap (
            dynamic_bitset< _WordT, _Alloc > & __b)  [inline], [noexcept]
```

Swap with another bitset.

References dynamic_bitset().

test()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
bool std::tr2::dynamic_bitset< _WordT, _Alloc >::test (
            size_type __pos) const  [inline]
```

Tests the value of a bit.

Parameters

__pos	The index of a bit.

Returns

The value at *__pos*.

Exceptions

std::out_of_range	If *__pos* is bigger the size of the set.

to_string()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
template<typename _CharT = char, typename _Traits = std::char_traits<_CharT>, typename _Alloc1 =
std::allocator<_CharT>>
std::basic_string< _CharT, _Traits, _Alloc1 > std::tr2::dynamic_bitset< _WordT, _Alloc >::to_↩
string (
            _CharT __zero = _CharT('0'),
            _CharT __one = _CharT('1')) const  [inline]
```

Returns a character interpretation of the dynamic_bitset.

Returns

The string equivalent of the bits.

Note the ordering of the bits: decreasing character positions correspond to increasing bit positions (see the main class notes for an example).

to_ullong()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
unsigned long long std::tr2::dynamic_bitset< _WordT, _Alloc >::to_ullong () const  [inline]
```

Returns a numerical interpretation of the dynamic_bitset.

Returns

The integral equivalent of the bits.

Exceptions

std::overflow_error	If there are too many bits to be represented in an `unsigned long`.

to_ulong()

```
template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
unsigned long std::tr2::dynamic_bitset< _WordT, _Alloc >::to_ulong () const  [inline]
```

Returns a numerical interpretation of the dynamic_bitset.

Returns

The integral equivalent of the bits.

Exceptions

std::overflow_error	If there are too many bits to be represented in an `unsigned long`.

The documentation for this class was generated from the following files:

- dynamic_bitset
- dynamic_bitset.tcc

6.414 std::ranges::elements_of< _Range, _Alloc > Struct Template Reference

`#include <elements_of.h>`

Public Attributes

- _Alloc **allocator**
- _Range **range**

6.414.1 Detailed Description

template<range _Range, typename _Alloc = allocator<byte>>
struct std::ranges::elements_of< _Range, _Alloc >

Since

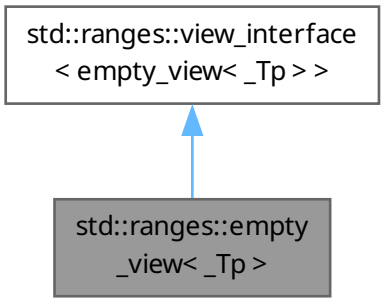
C++23

The documentation for this struct was generated from the following file:

- elements_of.h

6.415 std::ranges::empty_view< _Tp > Class Template Reference

`#include <ranges>`
Inheritance diagram for std::ranges::empty_view< _Tp >:



Public Member Functions

- constexpr decltype(auto) **back** ()
- constexpr auto **cbegin** ()
- constexpr auto **cend** ()
- constexpr decltype(auto) **front** ()
- constexpr **operator bool** () noexcept(noexcept(ranges::empty(_M_derived())))
- constexpr decltype(auto) **operator[ ]** (range_difference_t< _Range > __n)

Static Public Member Functions

- static constexpr _Tp * **begin** () noexcept
- static constexpr _Tp * **data** () noexcept
- static constexpr bool **empty** () noexcept
- static constexpr _Tp * **end** () noexcept
- static constexpr size_t **size** () noexcept

6.415.1 Detailed Description

template<typename _Tp>
requires is_object_v<_Tp>
class std::ranges::empty_view< _Tp >

A view that contains no elements.
The documentation for this class was generated from the following file:

- ranges

6.416 std::enable_if< bool, _Tp > Struct Template Reference

`#include <type_traits>`

6.416.1 Detailed Description

template<bool, typename _Tp = void>
struct std::enable_if< bool, _Tp >

Define a member typedef `type` only if a boolean constant is true.
The documentation for this struct was generated from the following file:

- type_traits

6.417 std::enable_shared_from_this< _Tp > Class Template Reference

`#include <memory>`

Public Member Functions

- shared_ptr< _Tp > **shared_from_this** ()
- shared_ptr< const _Tp > **shared_from_this** () const

Protected Member Functions

- **enable_shared_from_this** (const enable_shared_from_this &) noexcept
- enable_shared_from_this & **operator=** (const enable_shared_from_this &) noexcept

Friends

- const enable_shared_from_this * **__enable_shared_from_this_base** (const __shared_count<> &, const enable_shared_from_this *__p)
- template<typename, _Lock_policy>
  class **__shared_ptr**

6.417.1 Detailed Description

template<typename _Tp>
class std::enable_shared_from_this< _Tp >

Base class allowing use of the member function `shared_from_this`.

Since

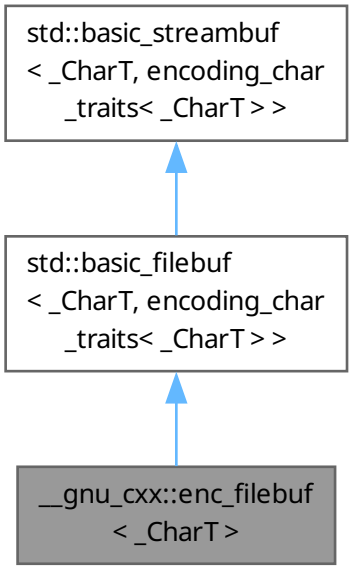
C++11

The documentation for this class was generated from the following file:

- bits/shared_ptr.h

6.418 __gnu_cxx::enc_filebuf< _CharT > Class Template Reference

`#include <enc_filebuf.h>`
Inheritance diagram for __gnu_cxx::enc_filebuf< _CharT >:



Public Types

- typedef codecvt< char_type, char, __state_type > **__codecvt_type**
- typedef __basic_file< char > **__file_type**
- typedef basic_filebuf< char_type, traits_type > **__filebuf_type**
- typedef traits_type::state_type **__state_type**
- typedef basic_streambuf< char_type, traits_type > **__streambuf_type**
- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**

- typedef traits_type::pos_type **pos_type**
- typedef traits_type::state_type **state_type**
- typedef encoding_char_traits< _CharT > **traits_type**

Public Member Functions

- **enc_filebuf** (state_type &__state)
- __filebuf_type * close ()
- locale getloc () const
- streamsize in_avail ()
- bool is_open () const throw ()
- __filebuf_type * open (const char *__s, ios_base::openmode __mode)
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)
- int_type sungetc ()
- void **swap** (basic_filebuf &)

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void **_M_allocate_internal_buffer** ()
- bool **_M_convert_to_external** (char_type *, streamsize)
- void _M_create_pback ()
- void **_M_destroy_internal_buffer** () throw ()
- void _M_destroy_pback () throw ()
- int **_M_get_ext_pos** (__state_type &__state)
- pos_type **_M_seek** (off_type __off, ios_base::seekdir __way, __state_type __state)
- void _M_set_buffer (streamsize __off)
- bool **_M_terminate_output** ()
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=encoding_char_traits< _CharT >::eof())
- virtual int_type pbackfail (int_type __c=encoding_char_traits< _CharT >::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual __streambuf_type * setbuf (char_type *__s, streamsize __n)

- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)

- char_type * eback () const
- char_type * gptr () const
- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- char_type * _M_buf
- bool **_M_buf_allocated**
- locale _M_buf_locale
- size_t _M_buf_size
- const __codecvt_type * **_M_codecvt**
- char * _M_ext_buf
- streamsize _M_ext_buf_size
- char * **_M_ext_end**
- const char * _M_ext_next
- __file_type **_M_file**
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- __c_lock **_M_lock**
- ios_base::openmode _M_mode
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end
- bool _M_reading
- __state_type **_M_state_beg**
- __state_type **_M_state_cur**
- __state_type **_M_state_last**
- bool **_M_writing**

- char_type _M_pback
- char_type * _M_pback_cur_save
- char_type * _M_pback_end_save
- bool _M_pback_init

6.418.1 Detailed Description

template<typename _CharT>
class __gnu_cxx::enc_filebuf< _CharT >

class enc_filebuf.

6.418.2 Member Function Documentation

_M_create_pback()

```
void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_create_pback ()  [inline],
[protected], [inherited]
```

Initializes pback buffers, and moves normal buffers to safety. Assumptions: _M_in_cur has already been moved back

_M_destroy_pback()

```
void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_destroy_pback () throw ( )
[inline], [protected], [inherited]
```

Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.

_M_set_buffer()

```
void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_set_buffer (
            streamsize __off)  [inline], [protected], [inherited]
```

This function sets the pointers of the internal buffer, both get and put areas. Typically:
__off == egptr() - eback() upon underflow/uflow (**read** mode); __off == 0 upon overflow (**write** mode); __off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).
NB: epptr() - pbase() == _M_buf_size - 1, since _M_buf_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.
Referenced by seekpos(), and setbuf().

close()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::__filebuf_type * std::basic_filebuf< ↩
_CharT, encoding_char_traits< _CharT > >::close ()  [inherited]
```

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.
If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.
If any operations fail, this function also fails.

eback()

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::eback () const  [inline],
[protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

egptr()

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::egptr () const  [inline],
[protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

epptr()

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::epptr () const  [inline],
[protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

gbump()

```
void std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::gbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.

getloc()

```
locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::getloc () const  [inline],
[inherited]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.

gptr()

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::gptr () const  [inline],
[protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

imbue()

```
void std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::imbue (
            const locale & __loc)  [protected], [virtual], [inherited]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

in_avail()

```
streamsize std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::in_avail ()  [inline],
[inherited]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

is_open()

```
bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::is_open () const throw ( )
[inline], [nodiscard], [inherited]
```

Returns true if the external file is open.

open()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::__filebuf_type * std::basic_filebuf< ↩
_CharT, encoding_char_traits< _CharT > >::open (
            const char * __s,
            ios_base::openmode __mode)  [inherited]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *__s* using the flags given in *__mode*.
Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596)

ios_base Flag combination				stdio equivalent
binary	in	out	trunc	app	
		+		w
		+	+	a
			+	a
	+	+		w
	+			r
	+	+		r+
	+	+	+	w+
	+	+	+	a+
	+		+	a+
+		+		wb
+		+	+	ab
+			+	ab
+		+	+	wb
+	+			rb
+	+	+		r+b
+	+	+	+	w+b
+	+	+	+	a+b
+	+		+	a+b

overflow()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::int_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::overflow (
            int_type __c = _Traits::eof())  [protected], [virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

_↩ _c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually __c, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if __c is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.
References _M_ext_buf, and _M_ext_next.

pbackfail()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::int_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::pbackfail (
            int_type __c = _Traits::eof())  [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

Parameters

_↩ _c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.
References _M_ext_buf, _M_ext_next, eback(), and std::min().

pbase()

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pbase () const  [inline],
[protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

pbump()

```
void std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.

pptr()

```
char_type * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pptr () const  [inline],
[protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

pubimbue()

```
locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubimbue (
            const locale & __loc)  [inline], [inherited]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
pos_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubseekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__off	Offset.

__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
pos_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubseekpos (
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
basic_streambuf * std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubsetbuf (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
int std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::pubsync ()  [inline], [inherited]
```

Calls virtual sync function.

sbumpc()

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sbumpc ()  [inline],
[inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.

seekoff()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::pos_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::seekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [protected], [virtual],
[inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

seekpos()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::pos_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::seekpos (
            pos_type __pos,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [protected], [virtual],
[inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.
References _M_reading, and _M_set_buffer().

setbuf()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::__streambuf_type * std::basic_filebuf<
_CharT, encoding_char_traits< _CharT > >::setbuf (
            char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Manipulates the buffer.

Parameters

__s	Pointer to a buffer area.
__n	Size of *__s*.

Returns

`this`

If no file has been opened, and both *__s* and *__n* are zero, then the stream becomes unbuffered. Otherwise, *__s* is used as a buffer; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering for more.
Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.
References _M_reading, _M_set_buffer(), and gptr().

setg()

```
void std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::setg (
            char_type * __gbeg,
            char_type * __gnext,
            char_type * __gend)  [inline], [protected], [inherited]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

setp()

```
void std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::setp (
            char_type * __pbeg,
            char_type * __pend)  [inline], [protected], [inherited]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

sgetc()

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sgetc ()  [inline],
[inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.

sgetn()

```
streamsize std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sgetn (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for xsgetn.

Parameters

_↩ _s	A buffer area.
_↩ _n	A count.

Returns xsgetn(__s,__n). The effect is to fill __s[0] through __s[__n-1] with characters from the input sequence, if possible.

showmanyc()

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::showmanyc ()  [protected],
[virtual], [inherited]
```

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

snextc()

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::snextc ()  [inline],
[inherited]
```

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.

sputbackc()

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sputbackc (
            char_type __c)  [inline], [inherited]
```

Pushing characters back into the input stream.

Parameters

_↩ _c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but __c is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be __c.

sputc()

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sputc (
            char_type __c)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _c	A character to output.

Returns

__c, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.

sputn()

```
streamsize std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sputn (
            const char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _s	A buffer read area.
_↩ _n	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __s[0] through __s[__n-1] to the output sequence, if possible.

sungetc()

```
int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::sungetc ()  [inline],
[inherited]
```

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.

sync()

```
int std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::sync ()  [protected], [virtual],
[inherited]
```

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

uflow()

```
virtual int_type std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::uflow ()  [inline],
[protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.

underflow()

```
basic_filebuf< _CharT, encoding_char_traits< _CharT > >::int_type std::basic_filebuf< _CharT,
encoding_char_traits< _CharT > >::underflow ()  [protected], [virtual], [inherited]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

xsgetn()

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::xsgetn (
            char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character extraction.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __s[0] through __s[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __n characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

xsputn()

```
streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character insertion.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __s[0] through __s[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented from std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >.

6.418.3 Member Data Documentation

_M_buf

```
char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_buf  [protected],
[inherited]
```

Pointer to the beginning of internal buffer.

_M_buf_locale

```
locale std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_buf_locale  [protected],
[inherited]
```

Current locale setting.

_M_buf_size

size_t std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_buf_size [protected], [inherited]

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.

_M_ext_buf

char* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_buf [protected], [inherited]

Buffer for external characters. Used for input when codecvt::always_noconv() == false. When valid, this corresponds to eback().

Referenced by overflow(), and pbackfail().

_M_ext_buf_size

streamsize std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_buf_size [protected], [inherited]

Size of buffer held by _M_ext_buf.

_M_ext_next

const char* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_ext_next [protected], [inherited]

Pointers into the buffer held by _M_ext_buf that delimit a subsequence of bytes that have been read but not yet converted. When valid, _M_ext_next corresponds to egptr().

Referenced by overflow(), and pbackfail().

_M_in_beg

char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_beg [protected], [inherited]

Start of get area.

_M_in_cur

char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_cur [protected], [inherited]

Current read area.

_M_in_end

char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_in_end [protected], [inherited]

End of get area.

_M_mode

ios_base::openmode std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_mode [protected], [inherited]

Place to stash in || out || in | out settings for current filebuf.

_M_out_beg

char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_out_beg [protected], [inherited]

Start of put area.

_M_out_cur

char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_out_cur [protected], [inherited]

Current put area.

_M_out_end

char_type* std::basic_streambuf< _CharT, encoding_char_traits< _CharT > >::_M_out_end [protected], [inherited]

End of put area.

_M_pback

char_type std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

_M_pback_cur_save

char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_cur_save [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

_M_pback_end_save

char_type* std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_end_save [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

_M_pback_init

bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_pback_init [protected], [inherited]

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

_M_reading

bool std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >::_M_reading [protected], [inherited]

_M_reading == false && _M_writing == false for **uncommitted** mode; _M_reading == true for **read** mode; _M_writing == true for **write** mode;

NB: _M_reading == true && _M_writing == true is unused.

Referenced by seekpos(), and setbuf().

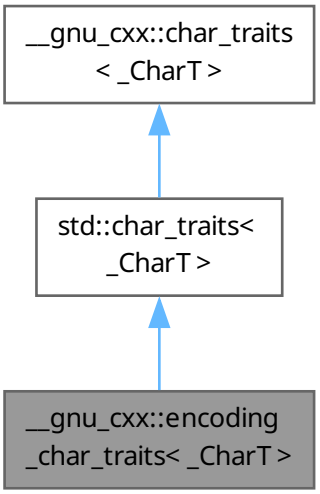
The documentation for this class was generated from the following file:

- enc_filebuf.h

6.419 __gnu_cxx::encoding_char_traits< _CharT > Struct Template Reference

#include <codecvt_specializations.h>

Inheritance diagram for __gnu_cxx::encoding_char_traits< _CharT >:



Public Types

- typedef _CharT **char_type**
- using **comparison_category**
- typedef _Char_types< _CharT >::int_type **int_type**
- typedef _Char_types< _CharT >::off_type **off_type**
- typedef std::fpos< state_type > **pos_type**
- typedef encoding_state **state_type**

Static Public Member Functions

- static constexpr void **assign** (char_type &__c1, const char_type &__c2)
- static constexpr char_type * **assign** (char_type *__s, std::size_t __n, char_type __a)
- static constexpr int **compare** (const char_type *__s1, const char_type *__s2, std::size_t __n)

- static constexpr char_type * **copy** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **eof** ()
- static constexpr bool **eq** (const char_type &__c1, const char_type &__c2)
- static constexpr bool **eq_int_type** (const int_type &__c1, const int_type &__c2)
- static constexpr const char_type * **find** (const char_type *__s, std::size_t __n, const char_type &__a)
- static constexpr std::size_t **length** (const char_type *__s)
- static constexpr bool **lt** (const char_type &__c1, const char_type &__c2)
- static constexpr char_type * **move** (char_type *__s1, const char_type *__s2, std::size_t __n)
- static constexpr int_type **not_eof** (const int_type &__c)
- static constexpr char_type **to_char_type** (const int_type &__c)
- static constexpr int_type **to_int_type** (const char_type &__c)

6.419.1 Detailed Description

template<**typename _CharT**>
struct __gnu_cxx::encoding_char_traits< **_CharT** >

encoding_char_traits
The documentation for this struct was generated from the following file:

- codecvt_specializations.h

6.420 __gnu_cxx::encoding_state Class Reference

`#include <codecvt_specializations.h>`

Public Types

- typedef iconv_t **descriptor_type**

Public Member Functions

- **encoding_state** (const char *__int, const char *__ext, int __ibom=0, int __ebom=0, int __bytes=1)
- **encoding_state** (const encoding_state &__obj)
- int **character_ratio** () const
- int **external_bom** () const
- const std::string **external_encoding** () const
- bool **good** () const throw ()
- const descriptor_type & **in_descriptor** () const
- int **internal_bom** () const
- const std::string **internal_encoding** () const
- encoding_state & **operator=** (const encoding_state &__obj)
- const descriptor_type & **out_descriptor** () const

Protected Member Functions

- void **construct** (const encoding_state &__obj)
- void **destroy** () throw ()
- void **init** ()

Protected Attributes

- int **_M_bytes**
- int **_M_ext_bom**
- std::string **_M_ext_enc**
- descriptor_type **_M_in_desc**
- int **_M_int_bom**
- std::string **_M_int_enc**
- descriptor_type **_M_out_desc**

6.420.1 Detailed Description

Extension to use iconv for dealing with character encodings.
The documentation for this class was generated from the following file:

- codecvt_specializations.h

6.421 std::encoding_state Class Reference

`#include <codecvt_specializations.h>`

Public Types

- typedef iconv_t **descriptor_type**

Public Member Functions

- **encoding_state** (const char *__int, const char *__ext, int __ibom=0, int __ebom=0, int __bytes=1)
- **encoding_state** (const encoding_state &__obj)
- int **character_ratio** () const
- int **external_bom** () const
- const std::string **external_encoding** () const
- bool **good** () const throw ()
- const descriptor_type & **in_descriptor** () const
- int **internal_bom** () const
- const std::string **internal_encoding** () const
- encoding_state & **operator=** (const encoding_state &__obj)
- const descriptor_type & **out_descriptor** () const

Protected Member Functions

- void **construct** (const encoding_state &__obj)
- void **destroy** () throw ()
- void **init** ()

Protected Attributes

- int **_M_bytes**
- int **_M_ext_bom**
- std::string **_M_ext_enc**
- descriptor_type **_M_in_desc**
- int **_M_int_bom**
- std::string **_M_int_enc**
- descriptor_type **_M_out_desc**

6.421.1 Detailed Description

Extension to use iconv for dealing with character encodings.
The documentation for this class was generated from the following file:

- codecvt_specializations.h

6.422 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw > Struct Template Reference

6.422.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw >

Entry compare, primary template.
The documentation for this struct was generated from the following file:

- entry_cmp.hpp

6.423 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false > Struct Template Reference

`#include <entry_cmp.hpp>`

Classes

- struct type

Public Types

- typedef __rebind_v::const_pointer **entry**

6.423.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >

Specialization, false.
The documentation for this struct was generated from the following file:

- entry_cmp.hpp

6.424 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true > Struct Template Reference

`#include <entry_cmp.hpp>`

Public Types

- typedef Cmp_Fn type

6.424.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >

Specialization, true.

6.424.2 Member Typedef Documentation

type

```
template<typename _VTp, typename Cmp_Fn, typename _Alloc>
typedef Cmp_Fn __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >::type
```

Compare.
The documentation for this struct was generated from the following file:

- entry_cmp.hpp

6.425 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw > Struct Template Reference

6.425.1 Detailed Description

template<typename _VTp, typename Pred, typename _Alloc, bool No_Throw>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw >

Entry predicate primary class template.
The documentation for this struct was generated from the following file:

- entry_pred.hpp

6.426 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false > Struct Template Reference

```
#include <entry_pred.hpp>
```

Public Types

- typedef __rebind_v::const_pointer **entry**

6.426.1 Detailed Description

template<typename _VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >

Specialization, false.
The documentation for this struct was generated from the following file:

- entry_pred.hpp

6.427 __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true > Struct Template Reference

```
#include <entry_pred.hpp>
```

Public Types

- typedef Pred **type**

6.427.1 Detailed Description

template<typename _VTp, typename Pred, typename _Alloc>
struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >

Specialization, true.
The documentation for this struct was generated from the following file:

- entry_pred.hpp

6.428 __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn > Struct Template Reference

`#include <eq_by_less.hpp>`

Public Member Functions

- bool **operator()** (const Key &r_lhs, const Key &r_rhs) const

6.428.1 Detailed Description

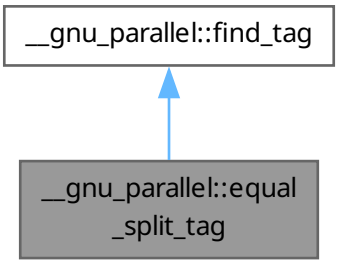
template<typename Key, class Cmp_Fn>
struct __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >

Equivalence function.
The documentation for this struct was generated from the following file:

- eq_by_less.hpp

6.429 __gnu_parallel::equal_split_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::equal_split_tag:



6.429.1 Detailed Description

Selects the equal splitting variant for std::find().

See also

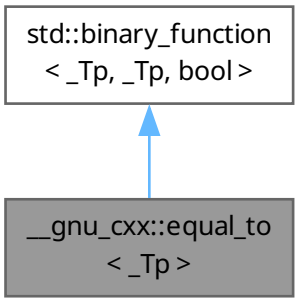
_GLIBCXX_FIND_EQUAL_SPLIT

The documentation for this struct was generated from the following file:

- tags.h

6.430 __gnu_cxx::equal_to< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for __gnu_cxx::equal_to< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

6.430.1 Detailed Description

template<typename _Tp>
struct __gnu_cxx::equal_to< _Tp >

One of the comparison functors.

6.430.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

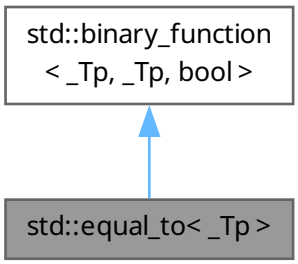
second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.431 std::equal_to< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::equal_to< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

6.431.1 Detailed Description

template<typename _Tp>
struct std::equal_to< _Tp >

One of the comparison functors.

6.431.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

second_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type  [inherited]
```

second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.432 std::ranges::equal_to Struct Reference

```
#include <ranges_cmp.h>
```

Public Types

- using **is_transparent**

Static Public Member Functions

- template<typename _Tp, typename _Up>
  static constexpr bool **operator()** (_Tp &&__t, _Up &&__u) noexcept(noexcept(std::declval< _Tp >()==std::declval< _Up >()))

6.432.1 Detailed Description

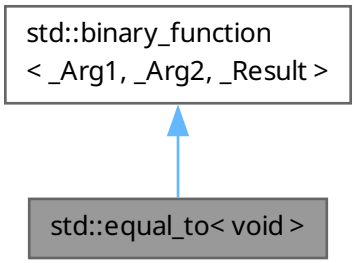
ranges::equal_to function object type.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

6.433 std::equal_to< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::equal_to< void >:



Public Types

- typedef void first_argument_type
- typedef void first_argument_type
- typedef __is_transparent **is_transparent**

- typedef bool result_type
- typedef bool result_type
- typedef void second_argument_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t)==std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t)==std::forward< _Up >(__u))
- constexpr bool **operator()** (const void &__x, const void &__y) const

6.433.1 Detailed Description

One of the comparison functors.

6.433.2 Member Typedef Documentation

first_argument_type `[1/2]`

```
typedef void std::binary_function< void, void, bool >::first_argument_type
```
`first_argument_type` is the type of the first argument

first_argument_type `[2/2]`

```
typedef void std::binary_function< void, void, bool >::first_argument_type
```
`first_argument_type` is the type of the first argument

result_type `[1/2]`

```
typedef bool std::binary_function< void, void, bool >::result_type
```
`result_type` is the return type

result_type `[2/2]`

```
typedef bool std::binary_function< void, void, bool >::result_type
```
`result_type` is the return type

second_argument_type `[1/2]`

```
typedef void std::binary_function< void, void, bool >::second_argument_type
```
`second_argument_type` is the type of the second argument

second_argument_type `[2/2]`

```
typedef void std::binary_function< void, void, bool >::second_argument_type
```
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.434 std::error_category Class Reference

```
#include <system_error>
```

Public Member Functions

- **error_category** (const error_category &)=delete
- virtual error_condition default_error_condition (int __i) const noexcept
- virtual bool equivalent (const error_code &__code, int __i) const noexcept
- virtual bool equivalent (int __i, const error_condition &__cond) const noexcept
- virtual string message (int) const =0
- virtual const char * name () const noexcept=0
- strong_ordering operator<=> (const error_category &__rhs) const noexcept
- error_category & **operator=** (const error_category &)=delete
- bool operator== (const error_category &__other) const noexcept

6.434.1 Detailed Description

Abstract base class for types defining a category of error codes.
An error category defines a context that gives meaning to the integer stored in an `error_code` or `error_condition` object. For example, the standard `errno` constants such a `EINVAL` and `ENOMEM` are associated with the "generic" category and other OS-specific error numbers are associated with the "system" category, but a user-defined category might give different meanings to the same numerical values.
A user-defined category can override the `equivalent` member functions to define correspondence between errors in different categories. For example, a category for errors from disk I/O could consider some of its error numbers equivalent to ENOSPC and ENOENT in the generic category.

Since

C++11

6.434.2 Member Function Documentation

default_error_condition()

```
virtual error_condition std::error_category::default_error_condition (
            int __i) const  [virtual], [noexcept]
```

Return an error_condition corresponding to `i` in this category.

equivalent() [1/2]

```
virtual bool std::error_category::equivalent (
            const error_code & __code,
            int __i) const  [virtual], [noexcept]
```

Test whether `code` corresponds to `i` for this category.

equivalent() [2/2]

```
virtual bool std::error_category::equivalent (
            int __i,
            const error_condition & __cond) const  [virtual], [noexcept]
```

Test whether `cond` corresponds to `i` for this category.

message()

```
virtual string std::error_category::message (
            int ) const  [pure virtual]
```

A description of the error condition corresponding to the number.
References message().
Referenced by message(), std::error_code::message(), and std::error_condition::message().

name()

```
virtual const char * std::error_category::name () const  [pure virtual], [noexcept]
```
A string that identifies the error category.

operator<=>()

```
strong_ordering std::error_category::operator<=> (
            const error_category & __rhs) const  [inline], [nodiscard], [noexcept]
```
Ordered comparison that defines a total order for error categories.
References operator<=>().
Referenced by operator<=>().

operator==()

```
bool std::error_category::operator== (
            const error_category & __other) const  [inline], [noexcept]
```
An error_category only compares equal to itself.
The documentation for this class was generated from the following file:

- system_error

6.435 std::error_code Class Reference

```
#include <system_error>
```

Public Member Functions

- template<typename _ErrorCodeEnum, typename = _Check<_ErrorCodeEnum>>
  error_code (_ErrorCodeEnum __e) noexcept
- **error_code** (const error_code &)=default
- **error_code** (int __v, const error_category &__cat) noexcept
- void **assign** (int __v, const error_category &__cat) noexcept
- const error_category & category () const noexcept
- void **clear** () noexcept
- error_condition default_error_condition () const noexcept
- string message () const
- operator bool () const noexcept
- error_code & **operator=** (const error_code &)=default
- int value () const noexcept

Related Symbols

(Note that these are not member symbols.)

- error_code make_error_code (errc __e) noexcept
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const error_code &__e)
- strong_ordering operator<=> (const error_code &__lhs, const error_code &__rhs) noexcept

6.435.1 Detailed Description

Class error_code
This class is a value type storing an integer error number and a category that gives meaning to the error number. Typically this is done close the the point where the error happens, to capture the original error value.
An `error_code` object can be used to store the original error value emitted by some subsystem, with a category relevant to the subsystem. For example, errors from POSIX library functions can be represented by an `errno` value and the "generic" category, but errors from an HTTP library might be represented by an HTTP response status code (e.g. 404) and a custom category defined by the library.

Since

C++11

6.435.2 Constructor & Destructor Documentation

error_code()

```
template<typename _ErrorCodeEnum, typename = _Check<_ErrorCodeEnum>>
std::error_code::error_code (
            _ErrorCodeEnum __e)  [inline], [noexcept]
```

Initialize with a user-defined type, by calling make_error_code.
References make_error_code().

6.435.3 Member Function Documentation

category()

```
const error_category & std::error_code::category () const  [inline], [noexcept]
```

The error category that this error belongs to.
Referenced by message().

default_error_condition()

```
error_condition std::error_code::default_error_condition () const  [noexcept]
```

An `error_condition` for this error's category and value.

message()

```
string std::error_code::message () const  [inline]
```

The category's description of the value.
References category(), std::error_category::message(), message(), and value().
Referenced by message().

operator bool()

```
std::error_code::operator bool () const  [inline], [explicit], [noexcept]
```

Test whether `value()` is non-zero.

value()

```
int std::error_code::value () const  [inline], [noexcept]
```

The error value.
Referenced by message().
The documentation for this class was generated from the following file:

- system_error

6.436 std::error_condition Class Reference

`#include <system_error>`

Public Member Functions

- error_condition () noexcept
- template<typename _ErrorConditionEnum, typename = _Check<_ErrorConditionEnum>>
  error_condition (_ErrorConditionEnum __e) noexcept
- **error_condition** (const error_condition &)=default
- error_condition (int __v, const error_category &__cat) noexcept
- void assign (int __v, const error_category &__cat) noexcept
- const error_category & category () const noexcept
- void clear () noexcept
- string message () const
- operator bool () const noexcept
- error_condition & **operator=** (const error_condition &)=default
- int value () const noexcept

Related Symbols

(Note that these are not member symbols.)

- error_condition make_error_condition (errc __e) noexcept
- strong_ordering operator<=> (const error_condition &__lhs, const error_condition &__rhs) noexcept
- bool operator== (const error_code &__lhs, const error_code &__rhs) noexcept
- bool operator== (const error_code &__lhs, const error_condition &__rhs) noexcept
- bool operator== (const error_condition &__lhs, const error_condition &__rhs) noexcept

6.436.1 Detailed Description

Class error_condition
This class represents error conditions that may be visible at an API boundary. Different `error_code` values that can occur within a library or module might map to the same `error_condition`.
An `error_condition` represents something that the program can test for, and subsequently take appropriate action.

Since

C++11

6.436.2 Constructor & Destructor Documentation

error_condition() [1/3]

```
std::error_condition::error_condition ()  [inline], [noexcept]
```

Initialize with a zero (no error) value and the generic category.
References std::generic_category().
Referenced by make_error_condition(), operator<=>(), operator==(), and operator==().

error_condition() [2/3]

```
std::error_condition::error_condition (
            int __v,
            const error_category & __cat)  [inline], [noexcept]
```

Initialize with the specified value and category.

error_condition() [3/3]

```
template<typename _ErrorConditionEnum, typename = _Check<_ErrorConditionEnum>>
std::error_condition::error_condition (
            _ErrorConditionEnum __e)  [inline], [noexcept]
```

Initialize with a user-defined type, by calling make_error_condition.
References make_error_condition().

6.436.3 Member Function Documentation

assign()

```
void std::error_condition::assign (
            int __v,
            const error_category & __cat)  [inline], [noexcept]
```

Set the value and category.
Referenced by clear().

category()

```
const error_category & std::error_condition::category () const  [inline], [noexcept]
```

The error category that this error belongs to.
Referenced by message().

clear()

```
void std::error_condition::clear ()  [inline], [noexcept]
```

Reset the value and category to the default-constructed state.
References assign(), and std::generic_category().

message()

```
string std::error_condition::message () const  [inline]
```

The category's description of the value.
References category(), std::error_category::message(), and value().

operator bool()

```
std::error_condition::operator bool () const  [inline], [explicit], [noexcept]
```

Test whether `value()` is non-zero.

value()

```
int std::error_condition::value () const  [inline], [noexcept]
```

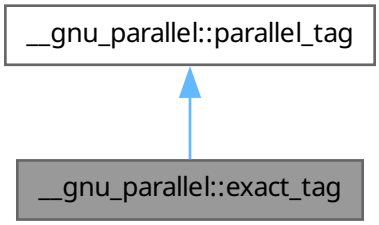
The error value.
Referenced by message().
The documentation for this class was generated from the following file:

- system_error

6.437 __gnu_parallel::exact_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::exact_tag:



Public Member Functions

- **exact_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

6.437.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

6.437.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

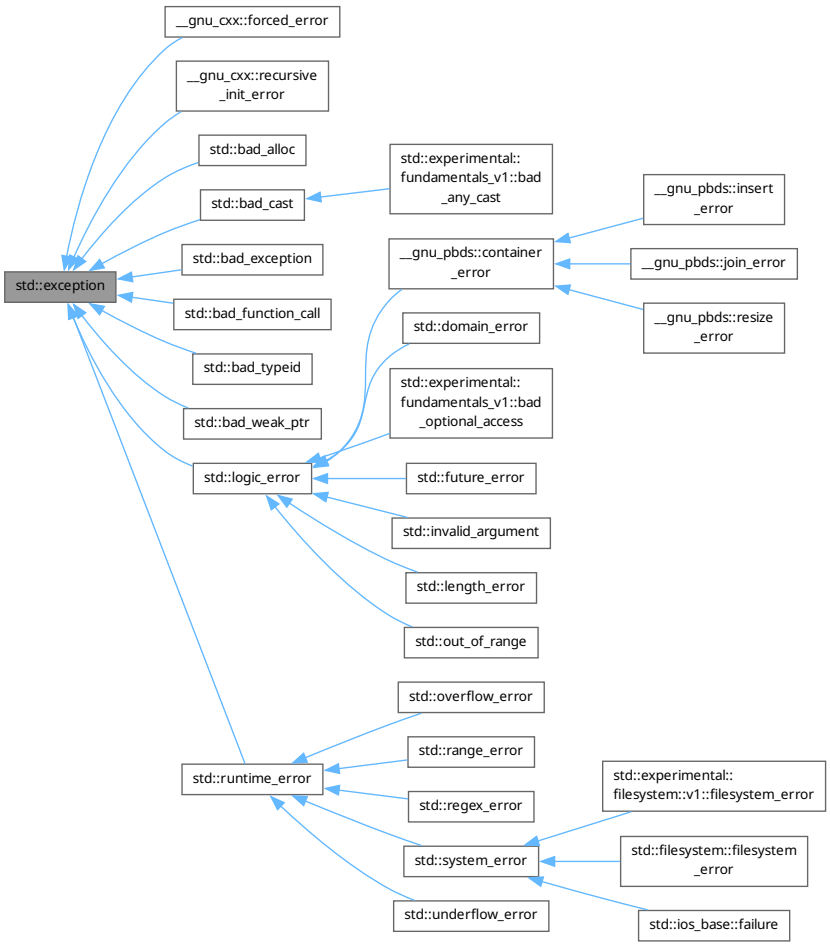
__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

6.438 std::exception Class Reference

`#include <exception.h>`
Inheritance diagram for std::exception:



Public Member Functions

- _GLIBCXX26_CONSTEXPR **exception** (const exception &)=default
- _GLIBCXX26_CONSTEXPR **exception** (exception &&)=default
- _GLIBCXX26_CONSTEXPR exception & **operator=** (const exception &)=default
- _GLIBCXX26_CONSTEXPR exception & **operator=** (exception &&)=default
- virtual constexpr const char * what () const noexcept

6.438.1 Detailed Description

Base class for all library exceptions.
This is the base class for all exceptions thrown by the standard library, and by certain language expressions. You are free to derive your own exception classes, or use a different hierarchy, or to throw non-class data (e.g., fundamental types).

6.438.2 Member Function Documentation

what()

```
virtual constexpr const char * std::exception::what () const  [inline], [constexpr], [virtual],
[noexcept]
```

Returns a C-style character string describing the general cause of the current error.
Reimplemented in std::bad_alloc, std::bad_cast, std::bad_exception, std::bad_function_call, std::bad_typeid, std::bad_weak_ptr, std::experimental::filesystem::v1::filesystem_error, std::experimental::fundamentals_v1::bad_any_cast, std::filesystem::filesystem_error, std::future_error, std::ios_base::failure, std::logic_error, and std::runtime_error.
The documentation for this class was generated from the following file:

- exception.h

6.439 std::__unspecified__::exception_ptr Class Reference

```
#include <exception>
```

Public Member Functions

- constexpr **exception_ptr** (const exception_ptr &) noexcept
- constexpr **exception_ptr** (exception_ptr &&__o) noexcept
- constexpr **exception_ptr** (nullptr_t) noexcept
- const class std::type_info * **__cxa_exception_type** () const noexcept
- constexpr **operator bool** () const noexcept
- constexpr exception_ptr & **operator=** (const exception_ptr &) noexcept
- constexpr exception_ptr & **operator=** (exception_ptr &&__o) noexcept
- constexpr void **swap** (exception_ptr &) noexcept

Friends

- constexpr friend bool **operator==** (const exception_ptr &, const exception_ptr &) noexcept=default
- exception_ptr **std::current_exception** () noexcept
- template<typename _Ex>
  constexpr exception_ptr **std::make_exception_ptr** (_Ex) noexcept
- void **std::rethrow_exception** (exception_ptr)
- constexpr void **swap** (exception_ptr &__lhs, exception_ptr &__rhs)

Related Symbols

(Note that these are not member symbols.)

- constexpr void **swap** (exception_ptr &__lhs, exception_ptr &__rhs)

6.439.1 Detailed Description

An opaque pointer to an arbitrary exception.
The actual name of this type is unspecified, so the alias `std::exception_ptr` should be used to refer to it.

Since

C++11 (but usable in C++98 as a GCC extension)

The documentation for this class was generated from the following file:

- exception_ptr.h

6.440 std::exception_ptr Class Reference

`#include <exception>`

Public Member Functions

- constexpr **exception_ptr** (const exception_ptr &) noexcept
- constexpr **exception_ptr** (exception_ptr &&__o) noexcept
- constexpr **exception_ptr** (nullptr_t) noexcept
- const class std::type_info * **__cxa_exception_type** () const noexcept
- constexpr **operator bool** () const noexcept
- constexpr exception_ptr & **operator=** (const exception_ptr &) noexcept
- constexpr exception_ptr & **operator=** (exception_ptr &&__o) noexcept
- constexpr void **swap** (exception_ptr &) noexcept

Friends

- constexpr friend bool **operator==** (const exception_ptr &, const exception_ptr &) noexcept=default
- exception_ptr **std::current_exception** () noexcept
- template<typename _Ex>
  constexpr exception_ptr **std::make_exception_ptr** (_Ex) noexcept
- void **std::rethrow_exception** (exception_ptr)
- constexpr void **swap** (exception_ptr &__lhs, exception_ptr &__rhs)

Related Symbols

(Note that these are not member symbols.)

- constexpr void **swap** (exception_ptr &__lhs, exception_ptr &__rhs)

6.440.1 Detailed Description

An opaque pointer to an arbitrary exception.
The actual name of this type is unspecified, so the alias `std::exception_ptr` should be used to refer to it.

Since

C++11 (but usable in C++98 as a GCC extension)

The documentation for this class was generated from the following file:

- exception_ptr.h

6.441 std::exponential_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- exponential_distribution ()
- exponential_distribution (_RealType __lambda)
- **exponential_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType lambda () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const exponential_distribution &__d1, const exponential_distribution &__d2)

6.441.1 Detailed Description

template<typename _RealType = double>
class std::exponential_distribution< _RealType >

An exponential continuous distribution for random numbers.
The formula for the exponential probability density function is $p(x|\lambda) = \lambda e^{-\lambda x}$.

Table 2305: Distribution Statistics

Mean	$\frac{1}{\lambda}$
Median	$\frac{\ln 2}{\lambda}$
Mode	$zero$
Range	$[0, \infty]$
Standard Deviation	$\frac{1}{\lambda}$

Since

C++11

6.441.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::exponential_distribution< _RealType >::result_type
```

The type of the range of the distribution.

6.441.3 Constructor & Destructor Documentation

exponential_distribution() [1/2]

```
template<typename _RealType = double>
std::exponential_distribution< _RealType >::exponential_distribution ()  [inline]
```

Constructs an exponential distribution with inverse scale parameter 1.0.

exponential_distribution() [2/2]

```
template<typename _RealType = double>
std::exponential_distribution< _RealType >::exponential_distribution (
            _RealType __lambda)  [inline], [explicit]
```

Constructs an exponential distribution with inverse scale parameter λ.

6.441.4 Member Function Documentation

lambda()

```
template<typename _RealType = double>
_RealType std::exponential_distribution< _RealType >::lambda () const  [inline]
```

Returns the inverse scale parameter of the distribution.

max()

```
template<typename _RealType = double>
result_type std::exponential_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::exponential_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::exponential_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::exponential_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

Referenced by std::operator>>().

param() [2/2]

```
template<typename _RealType = double>
void std::exponential_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::exponential_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.
Has no effect on exponential distributions.

6.441.5 Friends And Related Symbol Documentation

operator==

```
template<typename _RealType = double>
bool operator== (
            const exponential_distribution< _RealType > & __d1,
            const exponential_distribution< _RealType > & __d2)  [friend]
```

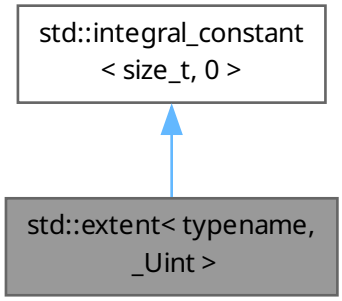
Return true if two exponential distributions have the same parameters.
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.442 std::extent< typename, _Uint > Struct Template Reference

```
#include <type_traits>
```

Inheritance diagram for std::extent< typename, _Uint >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr size_t **value**

6.442.1 Detailed Description

template<typename, unsigned _Uint = 0>
struct std::extent< typename, _Uint >

extent
The documentation for this struct was generated from the following file:

- type_traits

6.443 std::extreme_value_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **extreme_value_distribution** (_RealType __a, _RealType __b=_RealType(1))
- **extreme_value_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType a () const
- _RealType b () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const extreme_value_distribution &__d1, const extreme_value_distribution &__d2)

6.443.1 Detailed Description

template<typename _RealType = double>
class std::extreme_value_distribution< _RealType >

A extreme_value_distribution random number distribution.
The formula for the normal probability mass function is

$$p(x|a,b) = \frac{1}{b}\exp(\frac{a-x}{b} - \exp(\frac{a-x}{b}))$$

Since

C++11

6.443.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::extreme_value_distribution< _RealType >::result_type
```

The type of the range of the distribution.

6.443.3 Member Function Documentation

a()

```
template<typename _RealType = double>
_RealType std::extreme_value_distribution< _RealType >::a () const  [inline]
```

Return the a parameter of the distribution.

b()

```
template<typename _RealType = double>
_RealType std::extreme_value_distribution< _RealType >::b () const  [inline]
```

Return the b parameter of the distribution.

max()

```
template<typename _RealType = double>
result_type std::extreme_value_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::extreme_value_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::extreme_value_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::extreme_value_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
template<typename _RealType = double>
void std::extreme_value_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::extreme_value_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

6.443.4 Friends And Related Symbol Documentation

operator==

```
template<typename _RealType = double>
bool operator== (
            const extreme_value_distribution< _RealType > & __d1,
            const extreme_value_distribution< _RealType > & __d2)  [friend]
```

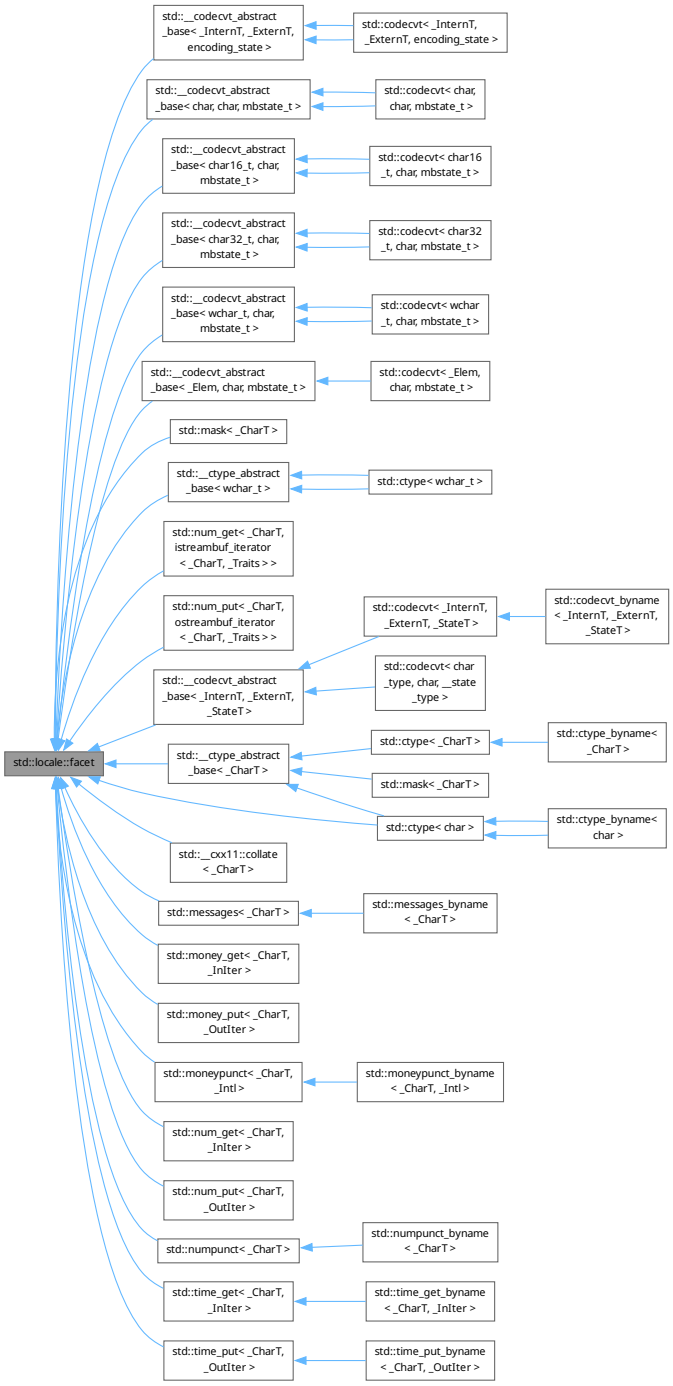
Return true if two extreme value distributions have the same parameters.
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.444 std::locale::facet Class Reference

```
#include <locale_classes.h>
```

Inheritance diagram for std::locale::facet:



Protected Member Functions

- **facet** (const facet &)=delete
- facet (size_t __refs=0) throw ()

- virtual ~facet ()
- facet & **operator=** (const facet &)=delete

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Friends

- class **locale**
- class **locale::_Impl**

6.444.1 Detailed Description

Localization functionality base class.
The facet class is the base class for a localization feature, such as money, time, and number printing. It provides common support for facets and reference management.
Facets may not be copied or assigned.

6.444.2 Constructor & Destructor Documentation

facet()

```
std::locale::facet::facet (
            size_t __refs = 0) throw ( )   [inline], [explicit], [protected]
```

Facet constructor.
This is the constructor provided by the standard. If refs is 0, the facet is destroyed when the last referencing locale is destroyed. Otherwise the facet will never be destroyed.

Parameters

__refs	The initial value for reference count.

Referenced by std::__cxx11::collate< _CharT >::collate(), std::__cxx11::collate< _CharT >::collate(), std::messages< _CharT >::messa
std::messages< _CharT >::messages(), std::money_get< _CharT, _InIter >::money_get(), std::money_put< _CharT, _OutIter >::money
std::moneypunct< _CharT, _Intl >::moneypunct(), std::moneypunct< _CharT, _Intl >::moneypunct(), std::moneypunct< _CharT, _Intl >:
std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::num_get(), std::num_put< _CharT, ostreambuf_iterator< _CharT, _Tr
std::numpunct< _CharT >::numpunct(), std::numpunct< _CharT >::numpunct(), std::numpunct< _CharT >::numpunct(),
std::time_get< _CharT, _InIter >::time_get(), std::time_put< _CharT, _OutIter >::time_put(), and ~facet().

~facet()

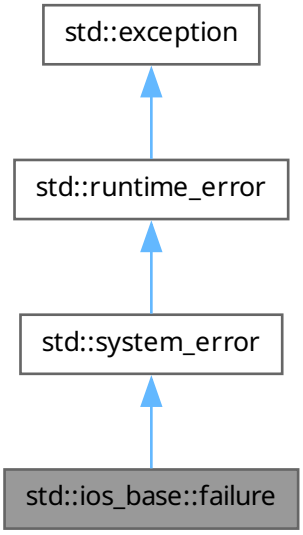
```
virtual std::locale::facet::~facet ()  [protected], [virtual]
```

Facet destructor.
References facet().
The documentation for this class was generated from the following file:

- locale_classes.h

6.445 std::ios_base::failure Class Reference

`#include <ios_base.h>`
Inheritance diagram for std::ios_base::failure:



Public Member Functions

- **failure** (const char *, const error_code &=io_errc::stream)
- **failure** (const string &, const error_code &)
- **failure** (const string &__str)
- const error_code & **code** () const noexcept
- virtual const char * what () const throw ()

6.445.1 Detailed Description

These are thrown to indicate problems with io.
27.4.2.1.1 Class ios_base::failure

6.445.2 Member Function Documentation

what()

`virtual const char * std::ios_base::failure::what () const throw ( )  [virtual]`
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::runtime_error.
References what().
Referenced by what().
The documentation for this class was generated from the following file:

- ios_base.h

6.446 std::filesystem::file_status Class Reference

`#include <filesystem>`

Public Member Functions

- **file_status** (const file_status &) noexcept=default
- **file_status** (file_status &&) noexcept=default
- **file_status** (file_type __ft, perms __prms=perms::unknown) noexcept
- file_status & **operator=** (const file_status &) noexcept=default
- file_status & **operator=** (file_status &&) noexcept=default
- perms **permissions** () const noexcept
- void **permissions** (perms __prms) noexcept
- file_type **type** () const noexcept
- void **type** (file_type __ft) noexcept

Friends

- bool **operator==** (const file_status &, const file_status &) noexcept=default

6.446.1 Detailed Description

Information about a file's type and permissions.

Since

C++17

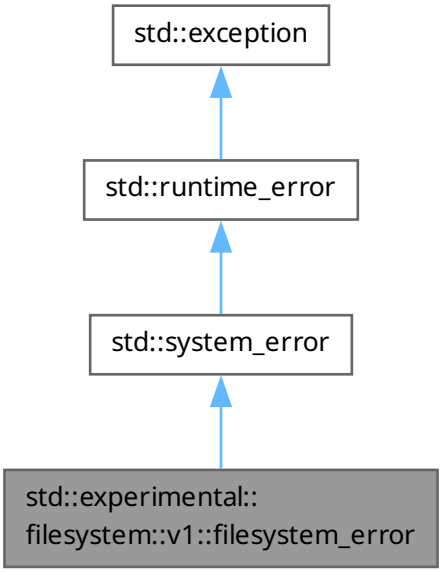
The documentation for this class was generated from the following file:

- bits/fs_dir.h

6.447 std::experimental::filesystem::v1::filesystem_error Class Reference

`#include <fs_path.h>`

Inheritance diagram for std::experimental::filesystem::v1::filesystem_error:



Public Member Functions

- **filesystem_error** (const string &__what_arg, const path &__p1, const path &__p2, error_code __ec)
- **filesystem_error** (const string &__what_arg, const path &__p1, error_code __ec)
- **filesystem_error** (const string &__what_arg, error_code __ec)
- const error_code & **code** () const noexcept
- const path & **path1** () const noexcept
- const path & **path2** () const noexcept
- const char * what () const noexcept

6.447.1 Detailed Description

Exception type thrown by the Filesystem TS library.

6.447.2 Member Function Documentation

what()

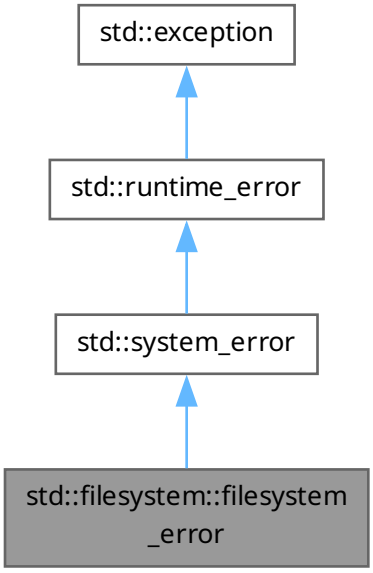
```
const char * std::experimental::filesystem::v1::filesystem_error::what () const  [inline], [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::runtime_error.
The documentation for this class was generated from the following file:

- experimental/bits/fs_path.h

6.448 std::filesystem::filesystem_error Class Reference

`#include <filesystem>`
Inheritance diagram for std::filesystem::filesystem_error:



Public Member Functions

- **filesystem_error** (const filesystem_error &)=default
- **filesystem_error** (const string &__what_arg, const path &__p1, const path &__p2, error_code __ec)
- **filesystem_error** (const string &__what_arg, const path &__p1, error_code __ec)
- **filesystem_error** (const string &__what_arg, error_code __ec)
- const error_code & **code** () const noexcept
- filesystem_error & **operator=** (const filesystem_error &)=default
- const path & **path1** () const noexcept
- const path & **path2** () const noexcept
- const char * what () const noexcept

6.448.1 Detailed Description

Exception type thrown by the Filesystem library.

Since

C++17

6.448.2 Member Function Documentation

what()

```
const char * std::filesystem::filesystem_error::what () const  [virtual], [noexcept]
```

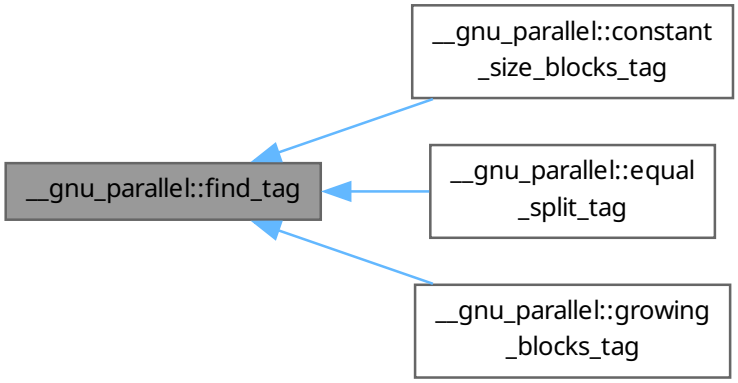
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::runtime_error.
References std::make_error_code().
The documentation for this class was generated from the following file:

- bits/fs_path.h

6.449 __gnu_parallel::find_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::find_tag:



6.449.1 Detailed Description

Base class for for std::find() variants.
The documentation for this struct was generated from the following file:

- tags.h

6.450 std::fisher_f_distribution< _RealType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **fisher_f_distribution** (_RealType __m, _RealType __n=_RealType(1))
- **fisher_f_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType **m** () const
- result_type max () const
- result_type min () const
- _RealType **n** () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::fisher_f_distribution< _RealType1 > &__x)
- bool operator== (const fisher_f_distribution &__d1, const fisher_f_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::fisher_f_distribution< _RealType1 > &__x)

6.450.1 Detailed Description

template<typename _RealType = double>
class std::fisher_f_distribution< _RealType >

A fisher_f_distribution random number distribution.
The formula for the normal probability mass function is

$$p(x|m,n) = \frac{\Gamma((m+n)/2)}{\Gamma(m/2)\Gamma(n/2)}(\frac{m}{n})^{m/2}x^{(m/2)-1}(1+\frac{mx}{n})^{-(m+n)/2}$$

Since

C++11

6.450.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::fisher_f_distribution< _RealType >::result_type
```

The type of the range of the distribution.

6.450.3 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::fisher_f_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::fisher_f_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::fisher_f_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.

param() [1/2]

```
template<typename _RealType = double>
param_type std::fisher_f_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::fisher_f_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::fisher_f_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

6.450.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::fisher_f_distribution< _RealType1 > & __x)  [friend]
```

Inserts a fisher_f_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A fisher_f_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
            const fisher_f_distribution< _RealType > & __d1,
            const fisher_f_distribution< _RealType > & __d2)  [friend]
```

Return true if two Fisher f distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::fisher_f_distribution< _RealType1 > & __x)  [friend]
```

Extracts a fisher_f_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A fisher_f_distribution random number generator engine.

Returns

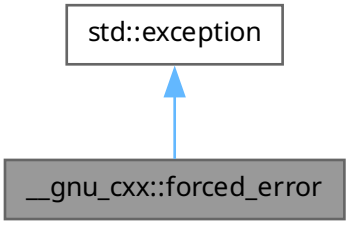
The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.451 __gnu_cxx::forced_error Struct Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::forced_error:



Public Member Functions

- virtual constexpr const char ∗ what () const noexcept

6.451.1 Detailed Description

Thrown by utilities for testing exception safety.

6.451.2 Member Function Documentation

what()

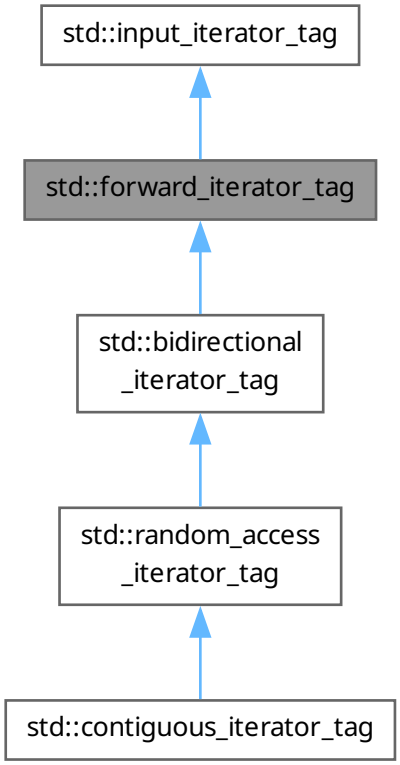
`virtual constexpr const char * std::exception::what () const [inline], [constexpr], [virtual], [noexcept], [inherited]`
Returns a C-style character string describing the general cause of the current error.
Reimplemented in std::bad_alloc, std::bad_cast, std::bad_exception, std::bad_function_call, std::bad_typeid, std::bad_weak_ptr, std::experimental::filesystem::v1::filesystem_error, std::experimental::fundamentals_v1::bad_any_cast, std::filesystem::filesystem_error, std::future_error, std::ios_base::failure, std::logic_error, and std::runtime_error.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.452 std::forward_iterator_tag Struct Reference

`#include <stl_iterator_base_types.h>`

Inheritance diagram for std::forward_iterator_tag:



6.452.1 Detailed Description

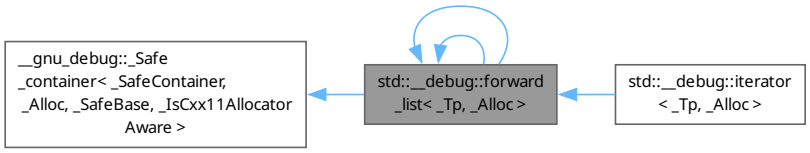
Forward iterators support a superset of input iterator operations.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

6.453 std::__debug::forward_list< _Tp, _Alloc > Class Template Reference

`#include <forward_list>`

Inheritance diagram for std::__debug::forward_list< _Tp, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, forward_list > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, forward_list > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **forward_list** (_Base_ref __x)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  forward_list (_InputIterator __first, _InputIterator __last, const allocator_type &__al=allocator_type())
- **forward_list** (const allocator_type &__al) noexcept
- **forward_list** (const forward_list &)=default
- **forward_list** (const forward_list &__list, const allocator_type &__al)
- **forward_list** (forward_list &&)=default
- **forward_list** (forward_list &&__list, const allocator_type &__al) noexcept(std::is_nothrow_constructible< _Base, _Base, const allocator_type & >::value)
- **forward_list** (size_type __n, const __type_identity_t< _Tp > &__value, const allocator_type &__al=allocator_type())
- **forward_list** (size_type __n, const allocator_type &__al=allocator_type())
- **forward_list** (std::initializer_list< _Tp > __il, const allocator_type &__al=allocator_type())
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__val)
- void **assign** (std::initializer_list< _Tp > __il)
- const_iterator **before_begin** () const noexcept
- iterator **before_begin** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbefore_begin** () const noexcept
- const_iterator **cbegin** () const noexcept

- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- template<typename... _Args>
  iterator **emplace_after** (const_iterator __pos, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- iterator **erase_after** (const_iterator __pos)
- iterator **erase_after** (const_iterator __pos, const_iterator __last)
- reference **front** ()
- const_reference **front** () const
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator **insert_after** (const_iterator __pos, _InputIterator __first, _InputIterator __last)
- iterator **insert_after** (const_iterator __pos, _Tp &&__val)
- iterator **insert_after** (const_iterator __pos, const _Tp &__val)
- iterator **insert_after** (const_iterator __pos, size_type __n, const _Tp &__val)
- iterator **insert_after** (const_iterator __pos, std::initializer_list< _Tp > __il)
- void **merge** (forward_list &&__list)
- template<typename _Comp>
  void **merge** (forward_list &&__list, _Comp __comp)
- void **merge** (forward_list &__list)
- template<typename _Comp>
  void **merge** (forward_list &__list, _Comp __comp)
- forward_list & **operator=** (const forward_list &)=default
- forward_list & **operator=** (forward_list &&)=default
- forward_list & **operator=** (std::initializer_list< _Tp > __il)
- void **pop_front** ()
- __remove_return_type **remove** (const _Tp &__val)
- template<typename _Pred>
  __remove_return_type **remove_if** (_Pred __pred)
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const value_type &__val)
- void **splice_after** (const_iterator __pos, forward_list &&__list)
- void **splice_after** (const_iterator __pos, forward_list &&__list, const_iterator __before, const_iterator __last)
- void **splice_after** (const_iterator __pos, forward_list &&__list, const_iterator __i)
- void **splice_after** (const_iterator __pos, forward_list &__list)
- void **splice_after** (const_iterator __pos, forward_list &__list, const_iterator __before, const_iterator __last)
- void **splice_after** (const_iterator __pos, forward_list &__list, const_iterator __i)
- void **swap** (forward_list &__list) noexcept(noexcept(declval< _Base & >().swap(__list)))
- __remove_return_type **unique** ()
- template<typename _BinPred>
  __remove_return_type **unique** (_BinPred __binary_pred)

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

6.453.1 Detailed Description

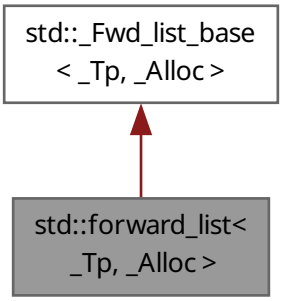
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::__debug::forward_list< _Tp, _Alloc >

Class std::forward_list with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/forward_list

6.454 std::forward_list< _Tp, _Alloc > Class Template Reference

`#include <forward_list>`
Inheritance diagram for std::forward_list< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _Base::const_iterator **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Base::iterator **iterator**
- typedef _Alloc_traits::pointer **pointer**
- typedef value_type & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- forward_list ()=default
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  forward_list (_InputIterator __first, _InputIterator __last, const _Alloc &__al=_Alloc())
- forward_list (const _Alloc &__al) noexcept
- forward_list (const forward_list &__list)
- forward_list (const forward_list &__list, const __type_identity_t< _Alloc > &__al)
- forward_list (forward_list &&)=default

- forward_list (forward_list &&__list, const __type_identity_t< _Alloc > &__al) noexcept(_Node_alloc_traits::_S↩ _always_equal())
- forward_list (size_type __n, const _Alloc &__al=_Alloc())
- forward_list (size_type __n, const _Tp &__value, const _Alloc &__al=_Alloc())
- forward_list (std::initializer_list< _Tp > __il, const _Alloc &__al=_Alloc())
- ~forward_list () noexcept
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  void assign (_InputIterator __first, _InputIterator __last)
- void assign (size_type __n, const _Tp &__val)
- void assign (std::initializer_list< _Tp > __il)
- const_iterator before_begin () const noexcept
- iterator before_begin () noexcept
- const_iterator begin () const noexcept
- iterator begin () noexcept
- const_iterator cbefore_begin () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- template<typename... _Args>
  iterator emplace_after (const_iterator __pos, _Args &&... __args)
- template<typename... _Args>
  reference emplace_front (_Args &&... __args)
- bool empty () const noexcept
- const_iterator end () const noexcept
- iterator end () noexcept
- iterator erase_after (const_iterator __pos)
- iterator erase_after (const_iterator __pos, const_iterator __last)
- reference front ()
- const_reference front () const
- allocator_type get_allocator () const noexcept
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator insert_after (const_iterator __pos, _InputIterator __first, _InputIterator __last)
- iterator **insert_after** (const_iterator __pos, _Tp &&__val)
- iterator insert_after (const_iterator __pos, const _Tp &__val)
- iterator insert_after (const_iterator __pos, size_type __n, const _Tp &__val)
- iterator insert_after (const_iterator __pos, std::initializer_list< _Tp > __il)
- size_type max_size () const noexcept
- void merge (forward_list &&__list)
- template<typename _Comp>
  void merge (forward_list &&__list, _Comp __comp)
- void **merge** (forward_list &__list)
- template<typename _Comp>
  void **merge** (forward_list &__list, _Comp __comp)
- forward_list & operator= (const forward_list &__list)
- forward_list & operator= (forward_list &&__list) noexcept(_Node_alloc_traits::_S_nothrow_move())
- forward_list & operator= (std::initializer_list< _Tp > __il)
- void pop_front ()
- void **push_front** (_Tp &&__val)
- void push_front (const _Tp &__val)
- __remove_return_type remove (const _Tp &__val)

- template<typename _Pred>
  __remove_return_type remove_if (_Pred __pred)
- void resize (size_type __sz)
- void resize (size_type __sz, const value_type &__val)
- void reverse () noexcept
- void sort ()
- template<typename _Comp>
  void sort (_Comp __comp)
- void splice_after (const_iterator __pos, forward_list &&__list) noexcept
- void splice_after (const_iterator __pos, forward_list &&__list, const_iterator __i) noexcept
- void **splice_after** (const_iterator __pos, forward_list &__list) noexcept
- void **splice_after** (const_iterator __pos, forward_list &__list, const_iterator __i) noexcept
- void swap (forward_list &__list) noexcept
- __remove_return_type unique ()
- template<typename _BinPred>
  __remove_return_type unique (_BinPred __binary_pred)

- void splice_after (const_iterator __pos, forward_list &&, const_iterator __before, const_iterator __last) noexcept
- void splice_after (const_iterator __pos, forward_list &, const_iterator __before, const_iterator __last) noexcept

6.454.1 Detailed Description

template<typename _Tp, typename _Alloc = allocator<_Tp>>
class std::forward_list< _Tp, _Alloc >

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Since

C++11

Template Parameters

_Tp	Type of element.
_Alloc	Allocator type, defaults to allocator<_Tp>.

Meets the requirements of a `container`, a `sequence`, including the `optional sequence requirements` with the exception of `at` and `operator[]`.
This is a *singly linked* list. Traversal up the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike std::vector and std::deque, random-access iterators are not provided, so subscripting (`[]`) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.
Also unlike the other standard containers, std::forward_list provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.

6.454.2 Constructor & Destructor Documentation

forward_list() `[1/10]`

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list ()  [default]
```

Creates a forward_list with no elements.
Referenced by insert_after(), insert_after(), merge(), operator=(), remove(), remove_if(), resize(), resize(), sort(), splice_after(), and unique().

forward_list() [2/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            const _Alloc & __al)  [inline], [explicit], [noexcept]
```

Creates a forward_list with no elements.

Parameters

_↩ _al	An allocator object.

forward_list() [3/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            const forward_list< _Tp, _Alloc > & __list,
            const __type_identity_t< _Alloc > & __al)  [inline]
```

Copy constructor with allocator argument.

Parameters

__list	Input list to copy.
__al	An allocator object.

forward_list() [4/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            forward_list< _Tp, _Alloc > && __list,
            const __type_identity_t< _Alloc > & __al)  [inline], [noexcept]
```

Move constructor with allocator argument.

Parameters

__list	Input list to move.
__al	An allocator object.

forward_list() [5/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            size_type __n,
            const _Alloc & __al = _Alloc())  [inline], [explicit]
```

Creates a forward_list with default constructed elements.

Parameters

_↩ _*n*	The number of elements to initially create.
_↩ _*al*	An allocator object.

This constructor creates the forward_list with ___n default constructed elements.

forward_list() [6/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            size_type __n,
            const _Tp & __value,
            const _Alloc & __al = _Alloc())  [inline]
```

Creates a forward_list with copies of an exemplar element.

Parameters

__n	The number of elements to initially create.
__value	An element to copy.
__al	An allocator object.

This constructor fills the forward_list with ___n copies of ___value.

forward_list() [7/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::forward_list< _Tp, _Alloc >::forward_list (
            _InputIterator __first,
            _InputIterator __last,
            const _Alloc & __al = _Alloc())  [inline]
```

Builds a forward_list from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__al	An allocator object.

Create a forward_list consisting of copies of the elements from [___first,___last). This is linear in N (where N is distance(___first,___last)).

forward_list() [8/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
```

```
            const forward_list< _Tp, _Alloc > & __list)  [inline]
```

The forward_list copy constructor.

Parameters

__list	A forward_list of identical element and allocator types.

forward_list() [9/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            forward_list< _Tp, _Alloc > && )  [default]
```

The forward_list move constructor.

Parameters

__list	A forward_list of identical element and allocator types.

The newly-created forward_list contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified forward_list.

forward_list() [10/10]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::forward_list (
            std::initializer_list< _Tp > __il,
            const _Alloc & __al = _Alloc())  [inline]
```

Builds a forward_list from an initializer_list.

Parameters

_↩ _il	An initializer_list of value_type.
_↩ _al	An allocator object.

Create a forward_list consisting of copies of the elements in the initializer_list __il. This is linear in __il.size().

~forward_list()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
std::forward_list< _Tp, _Alloc >::~forward_list ()  [inline], [noexcept]
```

The forward_list dtor.

6.454.3 Member Function Documentation

assign() [1/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::forward_list< _Tp, _Alloc >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Assigns a range to a forward_list.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a forward_list with copies of the elements in the range [__first,__last).
Note that the assignment completely changes the forward_list and that the number of elements of the resulting forward_list is the same as the number of elements assigned.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), operator=(), std::forward_list< _Tp, polymorphic_alloc and std::forward_list< _Tp, polymorphic_allocator< _Tp > >::operator=().

assign() [2/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::assign (
            size_type __n,
            const _Tp & __val)  [inline]
```

Assigns a given value to a forward_list.

Parameters

__n	Number of elements to be assigned.
__val	Value to be assigned.

This function fills a forward_list with __n copies of the given value. Note that the assignment completely changes the forward_list, and that the resulting forward_list has __n elements.

assign() [3/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::assign (
            std::initializer_list< _Tp > __il)  [inline]
```

Assigns an initializer_list to a forward_list.

Parameters

__il	An initializer_list of value_type.

Replace the contents of the forward_list with copies of the elements in the initializer_list __il. This is linear in __il.size().

before_begin() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::before_begin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points before the first element in the forward_list. Iteration is done in ordinary element order.

before_begin() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::before_begin ()  [inline], [noexcept]
```

Returns a read/write iterator that points before the first element in the forward_list. Iteration is done in ordinary element order.

Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), std::forward_list< _Tp, polymorphic_allocator< _Tp > insert_after(), insert_after(), resize(), and resize().

begin() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::begin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.

begin() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::begin ()  [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.

Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), std::forward_list< _Tp, polymorphic_allocator< _Tp > and unique().

cbefore_begin()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cbefore_begin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points before the first element in the forward_list. Iteration is done in ordinary element order.

Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), std::forward_list< _Tp, polymorphic_allocator< _Tp > std::forward_list< _Tp, polymorphic_allocator< _Tp > >::emplace_front(), std::forward_list< _Tp, polymorphic_allocator< _Tp > >::pus remove(), remove_if(), and unique().

cbegin()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the forward_list. Iteration is done in ordinary element order.

Referenced by std::operator==().

cend()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_iterator std::forward_list< _Tp, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

Referenced by std::operator==().

clear()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all the elements.
Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), std::forward_list< _Tp, polymorphic_allocator< _Tp > operator=(), and std::forward_list< _Tp, polymorphic_allocator< _Tp > >::operator=().

emplace_after()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename... _Args>
iterator std::forward_list< _Tp, _Alloc >::emplace_after (
            const_iterator __pos,
            _Args &&... __args)  [inline]
```

Constructs object in forward_list after the specified iterator.

Parameters

__pos	A const_iterator into the forward_list.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type `T` constructed with `T(std::forward<Args>(args)...)` after the specified location. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

emplace_front()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
template<typename... _Args>
reference std::forward_list< _Tp, _Alloc >::emplace_front (
            _Args &&... __args)  [inline]
```

Constructs object in forward_list at the front of the list.

Parameters

__args	Arguments.

This function will insert an object of type `Tp` constructed with `Tp(std::forward<Args>(args)...)` at the front of the list Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

empty()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
bool std::forward_list< _Tp, _Alloc >::empty () const  [inline], [noexcept]
```

Returns true if the forward_list is empty. (Thus begin() would equal end().)
Referenced by insert_after(), and sort().

end() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
```

```
const_iterator std::forward_list< _Tp, _Alloc >::end () const  [inline], [noexcept]
```
Returns a read-only iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.

end() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::end ()  [inline], [noexcept]
```
Returns a read/write iterator that points one past the last element in the forward_list. Iteration is done in ordinary element order.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), std::forward_list< _Tp, polymorphic_allocator< _Tp > insert_after(), insert_after(), resize(), resize(), and unique().

erase_after() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::erase_after (
            const_iterator __pos)  [inline]
```
Removes the element pointed to by the iterator following `pos`.

Parameters

__pos	Iterator pointing before element to be erased.

Returns

An iterator pointing to the element following the one that was erased, or `end()` if no such element exists.

This function will erase the element at the given position and thus shorten the forward_list by one.
Due to the nature of a forward_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), std::forward_list< _Tp, polymorphic_allocator< _Tp > resize(), and resize().

erase_after() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::erase_after (
            const_iterator __pos,
            const_iterator __last)  [inline]
```
Remove a range of elements.

Parameters

__pos	Iterator pointing before the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

`__last`

This function will erase the elements in the range `(__pos,__last)` and shorten the forward_list accordingly.
This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed.
The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

front() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
reference std::forward_list< _Tp, _Alloc >::front ()  [inline]
```

Returns a read/write reference to the data at the first element of the forward_list.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::emplace_front().

front() [2/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
const_reference std::forward_list< _Tp, _Alloc >::front () const  [inline]
```

Returns a read-only (constant) reference to the data at the first element of the forward_list.

get_allocator()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
allocator_type std::forward_list< _Tp, _Alloc >::get_allocator () const  [inline], [noexcept]
```

Get a copy of the memory allocation object.
Referenced by insert_after(), insert_after(), remove(), remove_if(), and unique().

insert_after() [1/4]

```
template<typename _Tp, typename _Alloc>
template<typename _InputIterator, typename>
forward_list< _Tp, _Alloc >::iterator forward_list::insert_after (
            const_iterator __pos,
            _InputIterator __first,
            _InputIterator __last)
```

Inserts a range into the forward_list.

Parameters

__pos	An iterator into the forward_list.
__first	An input iterator.
__last	An input iterator.

Returns

An iterator pointing to the last inserted element or __pos if `__first == __last`.

This function will insert copies of the data in the range `[ __first, __last)` into the forward_list after the location specified by `__pos`.
This operation is linear in the number of elements inserted and does not invalidate iterators and references.
References forward_list(), before_begin(), empty(), end(), and get_allocator().

insert_after() [2/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::insert_after (
            const_iterator __pos,
            const _Tp & __val)  [inline]
```

Inserts given value into forward_list after specified iterator.

Parameters

__pos	An iterator into the forward_list.
__val	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value after the specified location. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::assign(), std::forward_list< _Tp, polymorphic_allocator< _Tp > std::forward_list< _Tp, polymorphic_allocator< _Tp > >::insert_after(), and resize().

insert_after() [3/4]

```
template<typename _Tp, typename _Alloc>
forward_list< _Tp, _Alloc >::iterator forward_list::insert_after (
            const_iterator __pos,
            size_type __n,
            const _Tp & __val)
```

Inserts a number of copies of given data into the forward_list.

Parameters

__pos	An iterator into the forward_list.
__n	Number of elements to be inserted.
__val	Data to be inserted.

Returns

An iterator pointing to the last inserted copy of `val` or `pos` if `n == 0`.

This function will insert a specified number of copies of the given data after the location specified by `pos`.
This operation is linear in the number of elements inserted and does not invalidate iterators and references.
References forward_list(), before_begin(), end(), and get_allocator().

insert_after() [4/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
iterator std::forward_list< _Tp, _Alloc >::insert_after (
            const_iterator __pos,
            std::initializer_list< _Tp > __il)  [inline]
```

Inserts the contents of an initializer_list into forward_list after the specified iterator.

Parameters

__pos	An iterator into the forward_list.
__il	An initializer_list of value_type.

Returns

An iterator pointing to the last inserted element or __pos if __il is empty.

This function will insert copies of the data in the initializer_list __il into the forward_list before the location specified by __pos.
This operation is linear in the number of elements inserted and does not invalidate iterators and references.

max_size()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
size_type std::forward_list< _Tp, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the largest possible number of elements of forward_list.

merge() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::merge (
            forward_list< _Tp, _Alloc > && __list)  [inline]
```

Merge sorted lists.

Parameters

__list	Sorted list to merge.

Assumes that both __list and this list are sorted according to operator<(). Merges elements of __list into this list in sorted order, leaving __list empty when complete. Elements in this list precede elements in __list that are equal.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::merge().

merge() [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _Comp>
void forward_list::merge (
            forward_list< _Tp, _Alloc > && __list,
            _Comp __comp)
```

Merge sorted lists according to comparison function.

Parameters

__list	Sorted list to merge.
__comp	Comparison function defining sort order.

Assumes that both __list and this list are sorted according to comp. Merges elements of __list into this list in sorted order, leaving __list empty when complete. Elements in this list precede elements in __list that are equivalent according to comp().
References forward_list(), std::__addressof(), and std::move().

operator=() [1/3]

```
template<typename _Tp, typename _Alloc>
forward_list< _Tp, _Alloc > & forward_list::operator= (
            const forward_list< _Tp, _Alloc > & __list)
```

The forward_list assignment operator.

Parameters

__list	A forward_list of identical element and allocator types.

All the elements of `__list` are copied.
Whether the allocator is copied depends on the allocator traits.
References forward_list(), std::__addressof(), assign(), and clear().

operator=() [2/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list & std::forward_list< _Tp, _Alloc >::operator= (
            forward_list< _Tp, _Alloc > && __list)  [inline], [noexcept]
```

The forward_list move assignment operator.

Parameters

__list	A forward_list of identical element and allocator types.

The contents of `__list` are moved into this forward_list (without copying, if the allocators permit it).
Afterwards *__list* is a valid, but unspecified forward_list
Whether the allocator is moved depends on the allocator traits.

operator=() [3/3]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
forward_list & std::forward_list< _Tp, _Alloc >::operator= (
            std::initializer_list< _Tp > __il)  [inline]
```

The forward_list initializer list assignment operator.

Parameters

_↩ _il	An initializer_list of value_type.

Replace the contents of the forward_list with copies of the elements in the initializer_list `__il`. This is linear in `__↩ il.size()`.

pop_front()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::pop_front ()  [inline]
```

Removes first element.
This is a typical stack operation. It shrinks the forward_list by one. Due to the nature of a forward_list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.
Note that no data is returned, and if the first element's data is needed, it should be retrieved before `pop_front()` is called.

push_front()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::push_front (
            const _Tp & __val)  [inline]
```

Add data to the front of the forward_list.

Parameters

__val	Data to be added.

This is a typical stack operation. The function creates an element at the front of the forward_list and assigns the given data to it. Due to the nature of a forward_list this operation can be done in constant time, and does not invalidate iterators and references.

remove()

```
template<typename _Tp, typename _Alloc>
auto forward_list::remove (
            const _Tp & __val)
```

Remove all elements equal to value.

Parameters

__val	The value to remove.

Removes every element in the list equal to `__val`. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References forward_list(), cbefore_begin(), get_allocator(), and splice_after().

remove_if()

```
template<typename _Tp, typename _Alloc>
template<typename _Pred>
auto forward_list::remove_if (
            _Pred __pred) -> __remove_return_type
```

Remove all elements satisfying a predicate.

Parameters

__pred	Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References forward_list(), cbefore_begin(), get_allocator(), and splice_after().

resize() [1/2]

```
template<typename _Tp, typename _Alloc>
void forward_list::resize (
            size_type __sz)
```

Resizes the forward_list to the specified number of elements.

Parameters

__sz	Number of elements the forward_list should contain.

This function will resize the forward_list to the specified number of elements. If the number is smaller than the forward_list's current number of elements the forward_list is truncated, otherwise the forward_list is extended and the new elements are default constructed.
References forward_list(), before_begin(), end(), and erase_after().

resize() [2/2]

```
template<typename _Tp, typename _Alloc>
void forward_list::resize (
            size_type __sz,
            const value_type & __val)
```

Resizes the forward_list to the specified number of elements.

Parameters

__sz	Number of elements the forward_list should contain.
__val	Data with which new elements should be populated.

This function will resize the forward_list to the specified number of elements. If the number is smaller than the forward_list's current number of elements the forward_list is truncated, otherwise the forward_list is extended and new elements are populated with given data.
References forward_list(), before_begin(), end(), erase_after(), and insert_after().

reverse()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::reverse ()  [inline], [noexcept]
```

Reverse the elements in list.
Reverse the order of elements in the list in linear time.

sort() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::sort ()  [inline]
```

Sort the elements of the list.
Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.
Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::sort().

sort() [2/2]

```
template<typename _Tp, class _Alloc>
template<typename _Comp>
void forward_list::sort (
            _Comp __comp)
```

Sort the forward_list using a comparison function.
Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.
References forward_list(), and empty().

splice_after() [1/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
            const_iterator __pos,
            forward_list< _Tp, _Alloc > && ,
            const_iterator __before,
            const_iterator __last)  [inline], [noexcept]
```

Insert range from another forward_list.

Parameters

__pos	Iterator referencing the element to insert after.
__list	Source list.
__before	Iterator referencing before the start of range in __list.
__last	Iterator referencing the end of range in __list.

Removes elements in the range (__before,__last) and inserts them after __pos in constant time.
Undefined if __pos is in (__before,__last).

splice_after() [2/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
            const_iterator __pos,
            forward_list< _Tp, _Alloc > && __list)  [inline], [noexcept]
```

Insert contents of another forward_list.

Parameters

__pos	Iterator referencing the element to insert after.
__list	Source list.

The elements of list are inserted in constant time after the element referenced by pos. list becomes an empty list.
Requires this != &x.
Referenced by remove(), remove_if(), std::forward_list< _Tp, polymorphic_allocator< _Tp > >::splice_after(), and unique().

splice_after() [3/4]

```
template<typename _Tp, typename _Alloc>
void forward_list::splice_after (
            const_iterator __pos,
            forward_list< _Tp, _Alloc > && __list,
            const_iterator __i)  [noexcept]
```

Insert element from another forward_list.

Parameters

__pos	Iterator referencing the element to insert after.
__list	Source list.

__i	Iterator referencing the element before the element to move.

Removes the element in list __list referenced by __i and inserts it into the current list after __pos.
References forward_list().

splice_after() [4/4]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::splice_after (
            const_iterator __pos,
            forward_list< _Tp, _Alloc > & ,
            const_iterator __before,
            const_iterator __last)  [inline], [noexcept]
```

Insert range from another forward_list.

Parameters

__pos	Iterator referencing the element to insert after.
__list	Source list.
__before	Iterator referencing before the start of range in __list.
__last	Iterator referencing the end of range in __list.

Removes elements in the range (__before,__last) and inserts them after __pos in constant time.
Undefined if __pos is in (__before,__last).

swap()

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
void std::forward_list< _Tp, _Alloc >::swap (
            forward_list< _Tp, _Alloc > & __list)  [inline], [noexcept]
```

Swaps data with another forward_list.

Parameters

__list	A forward_list of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global std::swap() function is overloaded such that std::swap(l1, l2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.

unique() [1/2]

```
template<typename _Tp, typename _Alloc = allocator<_Tp>>
__remove_return_type std::forward_list< _Tp, _Alloc >::unique ()  [inline]
```

Remove consecutive duplicate elements.
For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by std::forward_list< _Tp, polymorphic_allocator< _Tp > >::unique().

unique() [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _BinPred>
auto forward_list::unique (
            _BinPred __binary_pred) -> __remove_return_type
```

Remove consecutive elements satisfying a predicate.

Parameters

__binary_pred	Binary predicate function or object.

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References forward_list(), begin(), cbefore_begin(), end(), get_allocator(), and splice_after().
The documentation for this class was generated from the following files:

- forward_list.h
- forward_list.tcc

6.455 std::fpos< _StateT > Class Template Reference

```
#include <postypes.h>
```

Public Member Functions

- **fpos** (const fpos &)=default
- fpos (streamoff __off)
- operator streamoff () const
- fpos operator+ (streamoff __off) const
- fpos & operator+= (streamoff __off)
- streamoff operator- (const fpos &__other) const
- fpos operator- (streamoff __off) const
- fpos & operator-= (streamoff __off)
- fpos & **operator=** (const fpos &)=default
- _StateT state () const
- void state (_StateT __st)

6.455.1 Detailed Description

template<typename _StateT>
class std::fpos< _StateT >

Class representing stream positions.
The standard places no requirements upon the template parameter StateT. In this implementation StateT must be DefaultConstructible, CopyConstructible and Assignable. The standard only requires that fpos should contain a member of type StateT. In this implementation it also contains an offset stored as a signed integer.

Parameters

StateT	Type passed to and returned from state().

6.455.2 Constructor & Destructor Documentation

fpos()

```
template<typename _StateT>
std::fpos< _StateT >::fpos (
            streamoff __off)  [inline]
```

Construct position from offset.

6.455.3 Member Function Documentation

operator streamoff()

```
template<typename _StateT>
std::fpos< _StateT >::operator streamoff () const  [inline]
```

Convert to streamoff.

operator+()

```
template<typename _StateT>
fpos std::fpos< _StateT >::operator+ (
            streamoff __off) const  [inline]
```

Add position and offset.

operator+=()

```
template<typename _StateT>
fpos & std::fpos< _StateT >::operator+= (
            streamoff __off)  [inline]
```

Add offset to this position.

operator-() [1/2]

```
template<typename _StateT>
streamoff std::fpos< _StateT >::operator- (
            const fpos< _StateT > & __other) const  [inline]
```

Subtract position to return offset.

operator-() [2/2]

```
template<typename _StateT>
fpos std::fpos< _StateT >::operator- (
            streamoff __off) const  [inline]
```

Subtract offset from position.

operator-=()

```
template<typename _StateT>
fpos & std::fpos< _StateT >::operator-= (
            streamoff __off)  [inline]
```

Subtract offset from this position.

state() [1/2]

```
template<typename _StateT>
_StateT std::fpos< _StateT >::state () const  [inline]
```

Return the last set value of *st*.

state() [2/2]

```
template<typename _StateT>
void std::fpos< _StateT >::state (
            _StateT __st)  [inline]
```

Remember the value of *st*.

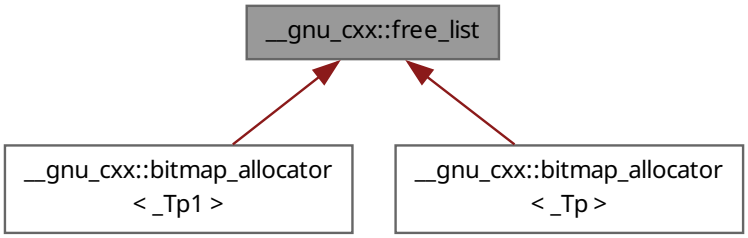
The documentation for this class was generated from the following file:

- postypes.h

6.456 __gnu_cxx::free_list Class Reference

`#include <bitmap_allocator.h>`

Inheritance diagram for __gnu_cxx::free_list:



Public Types

- typedef __mutex **__mutex_type**
- typedef vector_type::iterator **iterator**
- typedef std::size_t * **value_type**
- typedef __detail::__mini_vector< value_type > **vector_type**

Public Member Functions

- void _M_clear ()
- std::size_t * _M_get (std::size_t __sz)
- void _M_insert (std::size_t *__addr) throw ()

6.456.1 Detailed Description

The free list class for managing chunks of memory to be given to and returned by the bitmap_allocator.

6.456.2 Member Function Documentation

_M_clear()

```
void __gnu_cxx::free_list::_M_clear ()
```

This function just clears the internal Free List, and gives back all the memory to the OS.

_M_get()

```
std::size_t * __gnu_cxx::free_list::_M_get (
            std::size_t __sz)
```

This function gets a block of memory of the specified size from the free list.

Parameters

__sz	The size in bytes of the memory required.

Returns

A pointer to the new memory block of size at least equal to that requested.

_M_insert()

```
void __gnu_cxx::free_list::_M_insert (
            std::size_t * __addr) throw ( )   [inline]
```

This function returns the block of memory to the internal free list.

Parameters

__addr	The pointer to the memory block that was given by a call to the _M_get function.

Referenced by __gnu_cxx::bitmap_allocator< _Tp1 >::_M_deallocate_single_object().
The documentation for this class was generated from the following file:

- bitmap_allocator.h

6.457 std::from_chars_result Struct Reference

```
#include <charconv>
```

Public Member Functions

- constexpr **operator bool** () const noexcept

Public Attributes

- errc **ec**
- const char * **ptr**

Friends

- bool **operator==** (const from_chars_result &, const from_chars_result &)=default

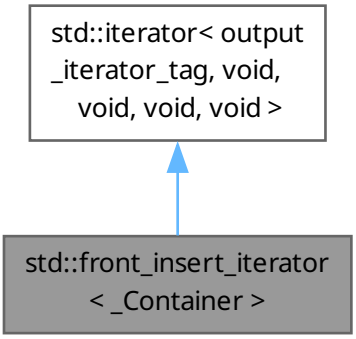
6.457.1 Detailed Description

Result type of std::from_chars.
The documentation for this struct was generated from the following file:

- charconv

6.458 std::front_insert_iterator< _Container > Class Template Reference

`#include <stl_iterator.h>`
Inheritance diagram for std::front_insert_iterator< _Container >:



Public Types

- typedef _Container container_type
- using **difference_type**
- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

Public Member Functions

- constexpr front_insert_iterator (_Container &__x)
- constexpr front_insert_iterator & operator* ()
- constexpr front_insert_iterator & operator++ ()
- constexpr front_insert_iterator operator++ (int)
- constexpr front_insert_iterator & operator= (const typename _Container::value_type &__value)
- constexpr front_insert_iterator & **operator=** (typename _Container::value_type &&__value)

Protected Attributes

- _Container * **container**

6.458.1 Detailed Description

template<typename _Container>
class std::front_insert_iterator< _Container >

Turns assignment into insertion.
These are output iterators, constructed from a container-of-T. Assigning a T to the iterator prepends it to the container using push_front.
Tip: Using the front_inserter function to create these iterators can save typing.

6.458.2 Member Typedef Documentation

container_type

```
template<typename _Container>
typedef _Container std::front_insert_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

iterator_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::↩
iterator_category  [inherited]
```

One of the tag types.

pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer  [inherited]
```

This type represents a pointer-to-value_type.

reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference  [inherited]
```

This type represents a reference-to-value_type.

value_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type  [inherited]
```

The type "pointed to" by the iterator.

6.458.3 Constructor & Destructor Documentation

front_insert_iterator()

```
template<typename _Container>
std::front_insert_iterator< _Container >::front_insert_iterator (
            _Container & __x)  [inline], [explicit], [constexpr]
```

The only way to create this iterator is with a container.
References std::__addressof().

6.458.4 Member Function Documentation

operator*()

```
template<typename _Container>
front_insert_iterator & std::front_insert_iterator< _Container >::operator* ()  [inline], [nodiscard],
[constexpr]
```

Simply returns *this.

operator++() [1/2]

```
template<typename _Container>
front_insert_iterator & std::front_insert_iterator< _Container >::operator++ ()  [inline], [constexpr]
```

Simply returns *this. (This iterator does not *move*.).

operator++() [2/2]

```
template<typename _Container>
front_insert_iterator std::front_insert_iterator< _Container >::operator++ (
            int )  [inline], [constexpr]
```

Simply returns *this. (This iterator does not *move*.).

operator=()

```
template<typename _Container>
front_insert_iterator & std::front_insert_iterator< _Container >::operator= (
            const typename _Container::value_type & __value)  [inline], [constexpr]
```

Parameters

__value	An instance of whatever type container_type::const_reference is; presumably a reference-to-const T for container<T>.

Returns

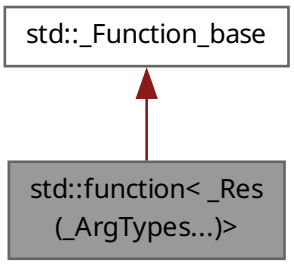
This iterator, for chained operations.

This kind of iterator doesn't really have a *position* in the container (you can think of the position as being permanently at the front, if you like). Assigning a value to the iterator will always prepend the value to the front of the container.
The documentation for this class was generated from the following file:

- bits/stl_iterator.h

6.459 std::function< _Res(_ArgTypes...)> Class Template Reference

`#include <std_function.h>`
Inheritance diagram for std::function< _Res(_ArgTypes...)>:



Public Types

- typedef _Res **result_type**

Public Member Functions

- function () noexcept
- template<typename _Functor, typename _Constraints = _Requires<_Callable<_Functor>>>
  function (_Functor &&__f) noexcept(_Handler< _Functor >::template _S_nothrow_init< _Functor >())
- function (const function &__x)
- function (function &&__x) noexcept
- function (nullptr_t) noexcept
- operator bool () const noexcept
- _Res operator() (_ArgTypes... __args) const
- template<typename _Functor>
  _Requires< _Callable< _Functor >, function & > operator= (_Functor &&__f) noexcept(_Handler< _Functor >::template _S_nothrow_init< _Functor >())
- function & operator= (const function &__x)
- function & operator= (function &&__x) noexcept
- function & operator= (nullptr_t) noexcept
- template<typename _Functor>
  function & operator= (reference_wrapper< _Functor > __f) noexcept
- void swap (function &__x) noexcept
- const type_info & target_type () const noexcept

- template<typename _Functor>
  _Functor * target () noexcept
- template<typename _Functor>
  const _Functor * target () const noexcept

6.459.1 Detailed Description

template<typename _Res, typename... _ArgTypes>
class std::function< _Res(_ArgTypes...)>

Polymorphic function wrapper.

Since

C++11

6.459.2 Constructor & Destructor Documentation

function() [1/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function () [inline], [noexcept]
```

Default construct creates an empty function call wrapper.

Postcondition

`!(bool)*this`

References function().
Referenced by function(), function(), function(), function(), function(), operator=(), operator=(), operator=(), operator=(), operator=(), swap(), and target().

function() [2/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
            nullptr_t ) [inline], [noexcept]
```

Creates an empty function call wrapper.

Postcondition

! (bool)*this

References function().

function() [3/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
            const function< _Res(_ArgTypes...)> & __x) [inline]
```

Function copy constructor.

Parameters

_↩ _x	A function object with identical call signature.

Postcondition

`bool(*this) == bool(__x)`

The newly-created function contains a copy of the target of `__x` (if it has one).
References function().

function() [4/5]

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::function (
            function< _Res(_ArgTypes...)> && __x) [inline], [noexcept]
```

Function move constructor.

Parameters

_↩ _x	A function object rvalue with identical call signature.

The newly-created function contains the target of `__x` (if it has one).
References function().

function() [5/5]

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor, typename _Constraints = _Requires<_Callable<_Functor>>>
std::function< _Res(_ArgTypes...)>::function (
            _Functor && __f) [inline], [noexcept]
```

Builds a function that targets a copy of the incoming function object.

Parameters

↩ _↩ ↩ _↩ *f*	A function object that is callable with parameters of type `ArgTypes...` and returns a value convertible to `Res`.

The newly-created function object will target a copy of `__f`. If `__f` is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If `__f` is a null function pointer, null pointer-to-member, or empty `std::function`, the newly-created object will be empty.
If `__f` is a non-null function pointer or an object of type `reference_wrapper<F>`, this function will not throw.
References function(), and std::forward().

6.459.3 Member Function Documentation

operator bool()

```
template<typename _Res, typename... _ArgTypes>
std::function< _Res(_ArgTypes...)>::operator bool () const [inline], [explicit], [noexcept]
```

Determine if the function wrapper has a target.

Returns

`true` when this function object contains a target, or `false` when it is empty.

This function will not throw exceptions.

operator()()

```
template<typename _Res, typename... _ArgTypes>
_Res std::function< _Res(_ArgTypes...)>::operator() (
            _ArgTypes... __args) const [inline]
```

Invokes the function targeted by `*this`.

Returns

the result of the target.

Exceptions

`bad_function_call`	when `!(bool)*this`

The function call operator invokes the target function object stored by `this`.
References std::forward(), and operator()().
Referenced by operator()().

operator=() [1/5]

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor>
_Requires< _Callable< _Functor >, function & > std::function< _Res(_ArgTypes...)>::operator= (
            _Functor && __f) [inline], [noexcept]
```

Function assignment to a new target.

Parameters

↩ _↩ ↩ _↩ *f*	A function object that is callable with parameters of type `_ArgTypes...` and returns a value convertible to `_Res`.

Returns

`*this`

Since

C++11

This function object wrapper will target a copy of `__f`. If `__f` is `reference_wrapper<F>`, then this function object will contain a reference to the function object `__f.get()`. If `__f` is a null function pointer or null pointer-to-member, this object will be empty.
If `__f` is a non-null function pointer or an object of type `reference_wrapper<F>`, this function will not throw.
References function(), std::forward(), and operator=().

operator=() [2/5]

```
template<typename _Res, typename... _ArgTypes>
function & std::function< _Res(_ArgTypes...)>::operator= (
            const function< _Res(_ArgTypes...)> & __x)  [inline]
```

Function assignment operator.

Parameters

_↩ *_x*	A function with identical call signature.

Postcondition

`(bool)*this == (bool)x`

Returns

`*this`

The target of `__x` is copied to `*this`. If `__x` has no target, then `*this` will be empty.
If `__x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.
References function(), and operator=().
Referenced by operator=(), operator=(), operator=(), operator=(), and operator=().

operator=() [3/5]

```
template<typename _Res, typename... _ArgTypes>
function & std::function< _Res(_ArgTypes...)>::operator= (
            function< _Res(_ArgTypes...)> && __x)  [inline], [noexcept]
```

Function move-assignment operator.

Parameters

_↩ _x	A function rvalue with identical call signature.

Returns

`*this`

The target of `__x` is moved to `*this`. If `__x` has no target, then `*this` will be empty.
If `__x` targets a function pointer or a reference to a function object, then this operation will not throw an exception.
References function(), std::move(), and operator=().

operator=() [4/5]

```
template<typename _Res, typename...  _ArgTypes>
function & std::function< _Res(_ArgTypes...)>::operator= (
            nullptr_t )  [inline], [noexcept]
```

Function assignment to empty.

Postcondition

`!(bool)*this`

Returns

`*this`

The target of `*this` is deallocated, leaving it empty.
References function(), and operator=().

operator=() [5/5]

```
template<typename _Res, typename...  _ArgTypes>
template<typename _Functor>
function & std::function< _Res(_ArgTypes...)>::operator= (
            reference_wrapper< _Functor > __f)  [inline], [noexcept]
```

This is an overloaded member function, provided for convenience. It differs from the above function only in what argument(s) it accepts.
References function(), and operator=().

swap()

```
template<typename _Res, typename...  _ArgTypes>
void std::function< _Res(_ArgTypes...)>::swap (
            function< _Res(_ArgTypes...)> & __x)  [inline], [noexcept]
```

Swap the targets of two function objects.

Parameters

_↩ _x	A function with identical call signature.

Swap the targets of `this` function object and `__f`. This function will not throw exceptions.
References function(), and swap().
Referenced by swap().

target() [1/2]

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor>
const _Functor * std::function< _Res(_ArgTypes...)>::target () const [inline], [noexcept]
```
Access the stored target function object.

Returns

Returns a pointer to the stored target function object, if `typeid(_Functor).equals(target_type())`; otherwise, a null pointer.

This function does not throw exceptions.
References target(), and target_type().

target() [2/2]

```
template<typename _Res, typename... _ArgTypes>
template<typename _Functor>
_Functor * std::function< _Res(_ArgTypes...)>::target () [inline], [noexcept]
```
Access the stored target function object.

Returns

Returns a pointer to the stored target function object, if `typeid(_Functor).equals(target_type())`; otherwise, a null pointer.

This function does not throw exceptions.
References function(), and target().
Referenced by target(), and target().

target_type()

```
template<typename _Res, typename... _ArgTypes>
const type_info & std::function< _Res(_ArgTypes...)>::target_type () const [inline], [noexcept]
```
Determine the type of the target of this function object wrapper.

Returns

the type identifier of the target function object, or `typeid(void)` if `!(bool)*this`.

This function will not throw exceptions.
References target_type().
Referenced by target(), and target_type().
The documentation for this class was generated from the following file:

- std_function.h

6.460 std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference

```
#include <functional>
```

Public Member Functions

- template<typename _Fn, typename _Vt = remove_reference_t<_Fn>>
  constexpr function_ref (_Fn &&__f) noexcept
- template<typename _Fn>
  function_ref (_Fn *__fn) noexcept

- template<auto __fn, typename _Fn>
  constexpr function_ref (constant_wrapper< __fn, _Fn >) noexcept
- template<auto __fn, typename _Fn, typename _Td>
  constexpr function_ref (constant_wrapper< __fn, _Fn >, _Td _GLIBCXX_MOF_CV *__ptr) noexcept
- template<auto __fn, typename _Fn, typename _Up, typename _Td = remove_reference_t<_Up>>
  constexpr function_ref (constant_wrapper< __fn, _Fn >, _Up &&__ref) noexcept
- _Res operator() (_ArgTypes... __args) const noexcept(_Noex)
- template<typename _Tp>
  function_ref & **operator=** (_Tp)=delete

Friends

- template<typename _Func>
  constexpr auto & **__polyfunc::__base_of** (_Func &) noexcept
- template<typename _Func>
  constexpr auto & **__polyfunc::__invoker_of** (_Func &) noexcept
- template<typename _Dst, typename _Src>
  consteval bool **__polyfunc::__is_funcref_assignable** () noexcept
- template<typename _Src, typename _Dst>
  consteval bool **__polyfunc::__is_invoker_convertible** () noexcept

6.460.1 Detailed Description

template<typename _Res, typename... _ArgTypes, bool _Noex>
class std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

Non-owning polymorphic function wrapper.

Since

C++26

The `std::function_ref` class template is a non-owning call wrapper, that refers to a bound object. Using function_ref outside of the lifetime of the bound object has undefined behavior.
It supports const-qualification and no-throw guarantees. The qualifications and exception-specification of the signature are respected when invoking the reference function.

6.460.2 Constructor & Destructor Documentation

function_ref() [1/5]

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
template<typename _Fn>
std::function_ref< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
            _Fn * __fn)  [inline], [noexcept]
```

Target and bound object is function pointed by parameter.

function_ref() [2/5]

```
template<typename _Res, typename...  _ArgTypes, bool _Noex>
template<typename _Fn, typename _Vt = remove_reference_t<_Fn>>
std::function_ref< _Res(_ArgTypes...)  _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
            _Fn && __f)  [inline], [constexpr], [noexcept]
```

Target and bound object is object referenced by parameter.
References std::addressof(), and std::forward().

function_ref() [3/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<auto __fn, typename _Fn>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
            constant_wrapper< __fn, _Fn > ) [inline], [constexpr], [noexcept]
```

Target object is __fn. There is no bound object.
References std::__invoke().

function_ref() [4/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<auto __fn, typename _Fn, typename _Up, typename _Td = remove_reference_t<_Up>>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
            constant_wrapper< __fn, _Fn > ,
            _Up && __ref) [inline], [constexpr], [noexcept]
```

Target object is equivalent to std::bind_front<_fn>(std::ref(__ref)). Bound object is object referenced by second parameter.
References std::addressof().

function_ref() [5/5]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<auto __fn, typename _Fn, typename _Td>
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::function_ref (
            constant_wrapper< __fn, _Fn > ,
            _Td _GLIBCXX_MOF_CV * __ptr) [inline], [constexpr], [noexcept]
```

Target object is equivalent to std::bind_front<_fn>(__ptr). Bound object is object pointed by second parameter (if any).

6.460.3 Member Function Documentation

operator()()

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
_Res std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator() (
            _ArgTypes... __args) const [inline], [noexcept]
```

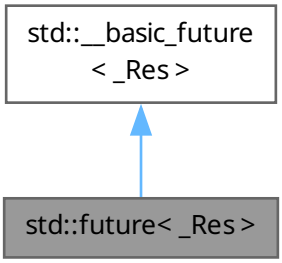
Invoke the target object.
The bound object will be invoked using the supplied arguments, and as const or non-const, as dictated by the template arguments of the `function_ref` specialization.
References std::forward().
The documentation for this class was generated from the following file:

- funcref_impl.h

6.461 std::future< _Res > Class Template Reference

```
#include <future>
```

Inheritance diagram for std::future< _Res >:



Public Member Functions

- **future** (const future &)=delete
- future (future &&__uf) noexcept
- _Res get ()
- future & **operator=** (const future &)=delete
- future & **operator=** (future &&__fut) noexcept
- shared_future< _Res > **share** () noexcept
- bool **valid** () const noexcept
- void **wait** () const
- template<typename _Rep, typename _Period>
  future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- template<typename _Clock, typename _Duration>
  future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< _Res > & **__result_type**

Protected Member Functions

- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

Friends

- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > **async** (launch, _Fn &&, _Args &&...)
- template<typename>
  class **packaged_task**
- class **promise< _Res >**

6.461.1 Detailed Description

template<typename _Res>
class std::future< _Res >

Primary template for future.

6.461.2 Constructor & Destructor Documentation

future()

```
template<typename _Res>
std::future< _Res >::future (
            future< _Res > && __uf)  [inline], [noexcept]
```

Move constructor.
References std::move().

6.461.3 Member Function Documentation

_M_get_result()

```
template<typename _Res>
__result_type std::__basic_future< _Res >::_M_get_result () const  [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.
Referenced by std::future< _Res >::get(), and std::shared_future< _Res >::get().

get()

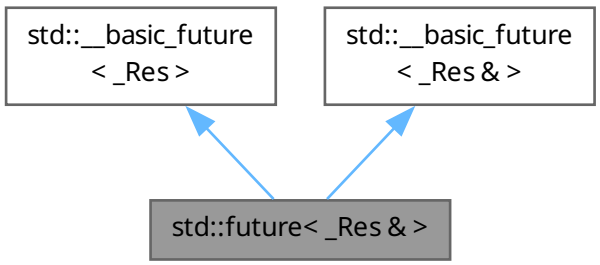
```
template<typename _Res>
_Res std::future< _Res >::get ()  [inline]
```

Retrieving the value.
References std::__basic_future< _Res >::_M_get_result(), and std::move().
The documentation for this class was generated from the following file:

- future

6.462 std::future< _Res & > Class Template Reference

`#include <future>`
Inheritance diagram for std::future< _Res & >:



Public Member Functions

- **future** (const future &)=delete
- future (future &&__uf) noexcept
- _Res & get ()
- future & **operator=** (const future &)=delete
- future & **operator=** (future &&__fut) noexcept
- shared_future< _Res & > **share** () noexcept
- bool **valid** () const noexcept
- void **wait** () const
- future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< _Res > & **__result_type**

Protected Member Functions

- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

Friends

- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > **async** (launch, _Fn &&, _Args &&...)
- template<typename>
  class **packaged_task**
- class **promise**< **_Res &** >
- class **promise**< **_Res** >

6.462.1 Detailed Description

template<typename _Res>
class std::future< _Res & >

Partial specialization for future<R&>.

6.462.2 Constructor & Destructor Documentation

future()

```
template<typename _Res>
std::future< _Res & >::future (
            future< _Res & > && __uf)  [inline], [noexcept]
```

Move constructor.
References std::move().

6.462.3 Member Function Documentation

_M_get_result()

```
__result_type std::__basic_future< _Res >::_M_get_result () const  [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.
Referenced by get().

get()

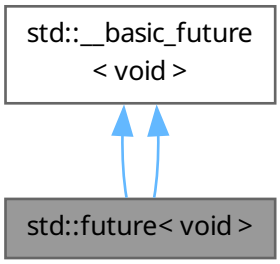
```
template<typename _Res>
_Res & std::future< _Res & >::get ()  [inline]
```

Retrieving the value.
References _M_get_result().
The documentation for this class was generated from the following file:

- future

6.463 std::future< void > Class Reference

`#include <future>`
Inheritance diagram for std::future< void >:



Public Member Functions

- **future** (const future &)=delete
- future (future &&__uf) noexcept
- void get ()
- future & **operator=** (const future &)=delete
- future & **operator=** (future &&__fut) noexcept
- shared_future< void > **share** () noexcept
- bool **valid** () const noexcept
- void **wait** () const
- future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< void > & **__result_type**

Protected Member Functions

- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

Friends

- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > **async** (launch, _Fn &&, _Args &&...)
- template<typename>
  class **packaged_task**
- class **promise**< **_Res** >
- class **promise**< **void** >

6.463.1 Detailed Description

Explicit specialization for future<void>.

6.463.2 Constructor & Destructor Documentation

future()

```
std::future< void >::future (
            future< void > && __uf)  [inline], [noexcept]
```

Move constructor.
References std::move().

6.463.3 Member Function Documentation

_M_get_result()

```
__result_type std::__basic_future< void >::_M_get_result () const  [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.
Referenced by get().

get()

```
void std::future< void >::get ()  [inline]
```

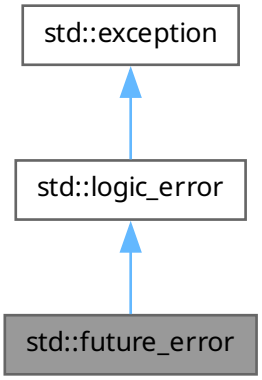
Retrieving the value.
References _M_get_result().
The documentation for this class was generated from the following file:

- future

6.464 std::future_error Class Reference

```
#include <future>
```

Inheritance diagram for std::future_error:



Public Member Functions

- **future_error** (future_errc __errc)
- const error_code & **code** () const noexcept
- virtual const char * what () const noexcept

Friends

- void **__throw_future_error** (int)

6.464.1 Detailed Description

Exception type thrown by futures.

Since

C++11

6.464.2 Member Function Documentation

what()

```
virtual const char * std::future_error::what () const  [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::logic_error.
The documentation for this class was generated from the following file:

- future

6.465 std::gamma_distribution< _RealType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- gamma_distribution ()
- gamma_distribution (_RealType __alpha_val, _RealType __beta_val=_RealType(1))
- **gamma_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType alpha () const
- _RealType beta () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::gamma_distribution< _RealType1 > &__x)
- bool operator== (const gamma_distribution &__d1, const gamma_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::gamma_distribution< _RealType1 > &__x)

6.465.1 Detailed Description

template<typename _RealType = double>
class std::gamma_distribution< _RealType >

A gamma continuous distribution for random numbers.
The formula for the gamma probability density function is:

$$p(x|\alpha,\beta) = \frac{1}{\beta\Gamma(\alpha)}(x/\beta)^{\alpha-1}e^{-x/\beta}$$

Since

C++11

6.465.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::gamma_distribution< _RealType >::result_type
```

The type of the range of the distribution.

6.465.3 Constructor & Destructor Documentation

gamma_distribution() [1/2]

```
template<typename _RealType = double>
std::gamma_distribution< _RealType >::gamma_distribution ()  [inline]
```

Constructs a gamma distribution with parameters 1 and 1.

gamma_distribution() [2/2]

```
template<typename _RealType = double>
std::gamma_distribution< _RealType >::gamma_distribution (
            _RealType __alpha_val,
            _RealType __beta_val = _RealType(1))  [inline], [explicit]
```

Constructs a gamma distribution with parameters α and β.

6.465.4 Member Function Documentation

alpha()

```
template<typename _RealType = double>
_RealType std::gamma_distribution< _RealType >::alpha () const  [inline]
```

Returns the α of the distribution.

beta()

```
template<typename _RealType = double>
_RealType std::gamma_distribution< _RealType >::beta () const  [inline]
```

Returns the β of the distribution.

max()

```
template<typename _RealType = double>
result_type std::gamma_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::gamma_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()() [1/2]

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::gamma_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

operator()() [2/2]

```
template<typename _RealType>
template<typename _UniformRandomNumberGenerator>
gamma_distribution< _RealType >::result_type std::gamma_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng,
            const param_type & __param)
```

Marsaglia, G. and Tsang, W. W. "A Simple Method for Generating Gamma Variables" ACM Transactions on Mathematical Software, 26, 3, 363-372, 2000.
References std::log(), and std::pow().

param() [1/2]

```
template<typename _RealType = double>
param_type std::gamma_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::gamma_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::gamma_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

6.465.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::gamma_distribution< _RealType1 > & __x)  [friend]
```

Inserts a gamma_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A gamma_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
            const gamma_distribution< _RealType > & __d1,
            const gamma_distribution< _RealType > & __d2)  [friend]
```

Return true if two gamma distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::gamma_distribution< _RealType1 > & __x)  [friend]
```

Extracts a gamma_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A gamma_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.466 std::geometric_distribution< _IntType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- **geometric_distribution** (const param_type &__p)
- **geometric_distribution** (double __p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- double p () const
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const geometric_distribution &__d1, const geometric_distribution &__d2)

6.466.1 Detailed Description

template<typename _IntType = int>
class std::geometric_distribution< _IntType >

A discrete geometric random number distribution.
The formula for the geometric probability density function is $p(i|p) = p(1-p)^i$ where p is the parameter of the distribution.

Since

C++11

6.466.2 Member Typedef Documentation

result_type

```
template<typename _IntType = int>
typedef _IntType std::geometric_distribution< _IntType >::result_type
```

The type of the range of the distribution.

6.466.3 Member Function Documentation

max()

```
template<typename _IntType = int>
result_type std::geometric_distribution< _IntType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _IntType = int>
result_type std::geometric_distribution< _IntType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::geometric_distribution< _IntType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

p()

```
template<typename _IntType = int>
double std::geometric_distribution< _IntType >::p () const  [inline]
```

Returns the distribution parameter p.

param() [1/2]

```
template<typename _IntType = int>
param_type std::geometric_distribution< _IntType >::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
template<typename _IntType = int>
void std::geometric_distribution< _IntType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _IntType = int>
void std::geometric_distribution< _IntType >::reset ()  [inline]
```

Resets the distribution state.
Does nothing for the geometric distribution.

6.466.4 Friends And Related Symbol Documentation

operator==

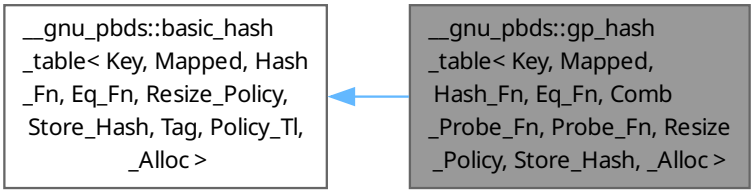
```
template<typename _IntType = int>
bool operator== (
            const geometric_distribution< _IntType > & __d1,
            const geometric_distribution< _IntType > & __d2)  [friend]
```

Return true if two geometric distributions have the same parameters.
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.467 __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc > Class Template Reference

`#include <assoc_container.hpp>`
Inheritance diagram for __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >:



Public Types

- typedef Comb_Probe_Fn **comb_probe_fn**
- typedef gp_hash_tag **container_category**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef Probe_Fn **probe_fn**
- typedef Resize_Policy **resize_policy**

Public Member Functions

- gp_hash_table ()
- **gp_hash_table** (const gp_hash_table &other)
- gp_hash_table (const hash_fn &h)
- gp_hash_table (const hash_fn &h, const eq_fn &e)
- gp_hash_table (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)
- gp_hash_table (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)
- gp_hash_table (const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p, const resize↩ _policy &rp)

- template<typename It>
  gp_hash_table (It first, It last)
- template<typename It>
  gp_hash_table (It first, It last, const hash_fn &h)
- template<typename It>
  gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e)
- template<typename It>
  gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp)
- template<typename It>
  gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p)
- template<typename It>
  gp_hash_table (It first, It last, const hash_fn &h, const eq_fn &e, const comb_probe_fn &cp, const probe_fn &p, const resize_policy &rp)
- gp_hash_table & **operator=** (const gp_hash_table &other)
- void **swap** (gp_hash_table &other)

6.467.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn = detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >

A general-probing hash-based associative container.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor.
Eq_Fn	Equal functor.
Comb_Probe_Fn	Combining probe functor. If Hash_Fn is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies.
Probe_Fn	Probe functor.
Resize_Policy	Resizes hash.
Store_Hash	Indicates whether the hash value will be stored along with each key. If Hash_Fn is null_type, then the container will not compile if this value is true
_Alloc	Allocator type.

Base tag choices are: gp_hash_tag.
Base is basic_hash_table.

6.467.2 Constructor & Destructor Documentation

gp_hash_table() [1/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
```

```
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↩
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↩
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table ()  [inline]
```

Default constructor.

gp_hash_table() [2/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↩
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↩
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
            const hash_fn & h)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object.

gp_hash_table() [3/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↩
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↩
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
            const hash_fn & h,
            const eq_fn & e)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, and r_eq_fn will be copied by the eq_fn object of the container object.

gp_hash_table() [4/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↩
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↩
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
            const hash_fn & h,
            const eq_fn & e,
            const comb_probe_fn & cp)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, and r_comb_probe_fn will be copied by the comb_probe_fn object of the container object.

gp_hash_table() [5/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↩
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↩
```

```
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
             const hash_fn & h,
             const eq_fn & e,
             const comb_probe_fn & cp,
             const probe_fn & p)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_probe_fn object of the container object, and r_probe_fn will be copied by the probe_fn object of the container object.

gp_hash_table() [6/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↩
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↩
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
             const hash_fn & h,
             const eq_fn & e,
             const comb_probe_fn & cp,
             const probe_fn & p,
             const resize_policy & rp)  [inline]
```

Constructor taking some policy objects. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_probe_fn object of the container object, r_probe_fn will be copied by the probe_fn object of the container object, and r_resize_policy will be copied by the Resize_Policy object of the container object.

gp_hash_table() [7/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↩
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↩
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
             It first,
             It last)  [inline]
```

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

gp_hash_table() [8/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↩
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↩
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
```

```
            It first,
            It last,
            const hash_fn & h)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object.

gp_hash_table() [9/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↩
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↩
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
            It first,
            It last,
            const hash_fn & h,
            const eq_fn & e)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, and r_eq_fn will be copied by the eq_fn object of the container object.

gp_hash_table() [10/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↩
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↩
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
            It first,
            It last,
            const hash_fn & h,
            const eq_fn & e,
            const comb_probe_fn & cp)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, and r_comb_probe_fn will be copied by the comb_probe_fn object of the container object.

gp_hash_table() [11/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↩
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↩
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
            It first,
```

```
            It last,
            const hash_fn & h,
            const eq_fn & e,
            const comb_probe_fn & cp,
            const probe_fn & p)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_probe_fn object of the container object, and r_probe_fn will be copied by the probe_fn object of the container object.

gp_hash_table() [12/12]

```
template<typename Key, typename Mapped, typename Hash_Fn = typename detail::default_hash_fn<↩
Key>::type, typename Eq_Fn = typename detail::default_eq_fn<Key>::type, typename Comb_Probe_Fn =
detail::default_comb_hash_fn::type, typename Probe_Fn = typename detail::default_probe_fn<Comb↩
_Probe_Fn>::type, typename Resize_Policy = typename detail::default_resize_policy<Comb_Probe_↩
Fn>::type, bool Store_Hash = detail::default_store_hash, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy,
Store_Hash, _Alloc >::gp_hash_table (
            It first,
            It last,
            const hash_fn & h,
            const eq_fn & e,
            const comb_probe_fn & cp,
            const probe_fn & p,
            const resize_policy & rp)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object. r_hash_fn will be copied by the hash_fn object of the container object, r_eq_fn will be copied by the eq_fn object of the container object, r_comb_probe_fn will be copied by the comb_↩ probe_fn object of the container object, r_probe_fn will be copied by the probe_fn object of the container object, and r_resize_policy will be copied by the resize_policy object of the container object.

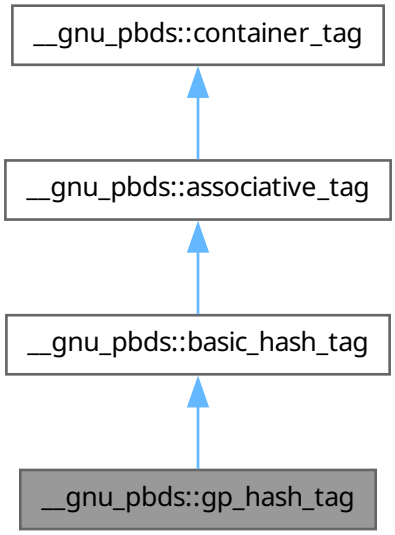
The documentation for this class was generated from the following file:

- assoc_container.hpp

6.468 __gnu_pbds::gp_hash_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::gp_hash_tag:



6.468.1 Detailed Description

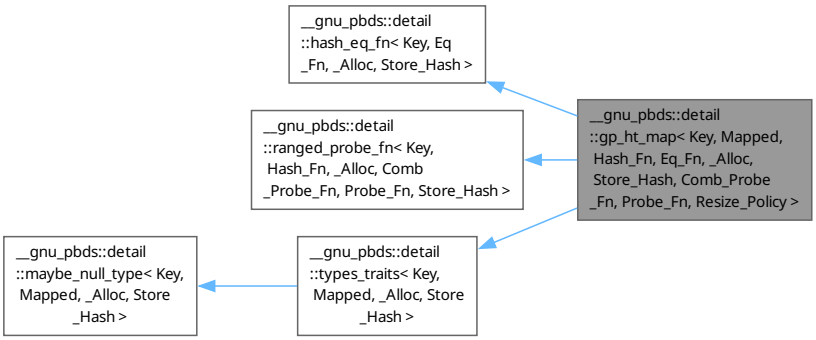
General-probing hash.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.469 __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > Class Template Reference

`#include <gp_ht_map_.hpp>`
Inheritance diagram for __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb↩

_Probe_Fn, Probe_Fn, Resize_Policy >:



Public Types

- enum { **store_hash** }
- typedef _Alloc **allocator_type**
- typedef Comb_Probe_Fn **comb_probe_fn**
- typedef const_iterator_ **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef Hash_Fn **hash_fn**
- typedef iterator_ **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef point_const_iterator_ **point_const_iterator**
- typedef point_iterator_ **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef Probe_Fn **probe_fn**
- typedef traits_base::reference **reference**
- typedef Resize_Policy **resize_policy**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **gp_ht_map** (const gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > &)
- **gp_ht_map** (const Hash_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Probe_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Probe_Fn &, const Probe_Fn &)
- **gp_ht_map** (const Hash_Fn &, const Eq_Fn &, const Comb_Probe_Fn &, const Probe_Fn &, const Resize_Policy &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It>
  void **copy_from_range** (It, It)
- bool empty () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- point_iterator **find_end** ()
- point_const_iterator **find_end** () const
- Comb_Probe_Fn & get_comb_probe_fn ()
- const Comb_Probe_Fn & get_comb_probe_fn () const
- Eq_Fn & get_eq_fn ()
- const Eq_Fn & get_eq_fn () const
- Hash_Fn & get_hash_fn ()
- const Hash_Fn & get_hash_fn () const
- Probe_Fn & get_probe_fn ()
- const Probe_Fn & get_probe_fn () const
- Resize_Policy & get_resize_policy ()
- const Resize_Policy & get_resize_policy () const
- std::pair< point_iterator, bool > **insert** (const_reference r_val)
- size_type **max_size** () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- size_type **size** () const
- void **swap** (gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy > &)

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Friends

- class **const_iterator_**
- class **iterator_**

6.469.1 Detailed Description

template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >

A general-probing hash-based container.

Template Parameters

Key	Key type.
Mapped	Map type.
Hash_Fn	Hashing functor. Default is __gnu_cxx::hash.
Eq_Fn	Equal functor. Default std::equal_to<Key>
_Alloc	Allocator type.
Store_Hash	If key type stores extra metadata. Defaults to false.
Comb_Probe_Fn	Combining probe functor. If Hash_Fn is not null_type, then this is the ranged-probe functor; otherwise, this is the range-hashing functor. XXX See Design::Hash-Based Containers::Hash Policies. Default direct_mask_range_hashing.
Probe_Fn	Probe functor. Defaults to linear_probe_fn, also quadratic_probe_fn.
Resize_Policy	Resizes hash. Defaults to hash_standard_resize_policy, using hash_exponential_size_policy and hash_load_check_resize_trigger.

Bases are: detail::hash_eq_fn, Resize_Policy, detail::ranged_probe_fn, detail::types_traits. (Optional: detail::debug_↩ map_base.)

6.469.2 Member Enumeration Documentation

anonymous enum

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
anonymous enum
```

Value stores hash, true or false.

6.469.3 Member Function Documentation

empty()

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
bool __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_↩
Fn, Probe_Fn, Resize_Policy >::empty () const  [inline], [nodiscard]
```

True if size() == 0.

get_comb_probe_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn ()
```

Return current comb_probe_fn.

get_comb_probe_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Comb_Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↩
Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_comb_probe_fn () const
```

Return current const comb_probe_fn.

get_eq_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩
Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn ()
```

Return current eq_fn.

get_eq_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Eq_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_eq_fn () const
```

Return current const eq_fn.

get_hash_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩
Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn ()
```

Return current hash_fn.

get_hash_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Hash_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_hash_fn () const
```

Return current const hash_fn.

get_probe_fn() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩
Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn ()
```

Return current probe_fn.

get_probe_fn() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Probe_Fn & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_probe_fn () const
```

Return current const probe_fn.

get_resize_policy() [1/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash,
Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy ()
```

Return current resize_policy.

get_resize_policy() [2/2]

```
template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool
Store_Hash, typename Comb_Probe_Fn, typename Probe_Fn, typename Resize_Policy>
const Resize_Policy & __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_↩
Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::get_resize_policy () const
```

Return current const resize_policy.
The documentation for this class was generated from the following file:

- gp_ht_map_.hpp

6.470 std::chrono::gps_clock Class Reference

`#include <chrono>`

Public Types

- using **duration**
- using **period**
- using **rep**
- using **time_point**

Static Public Member Functions

- template<typename _Duration>
  static gps_time< common_type_t< _Duration, seconds > > **from_utc** (const utc_time< _Duration > &__t)
- static time_point **now** ()
- template<typename _Duration>
  static utc_time< common_type_t< _Duration, seconds > > **to_utc** (const gps_time< _Duration > &__t)

Static Public Attributes

- static constexpr bool **is_steady**

6.470.1 Detailed Description

A clock that measures GPS time.
The epoch is 1980-01-06 00:00:00.

Since

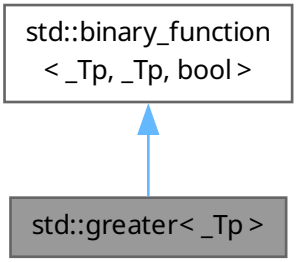
C++20

The documentation for this class was generated from the following file:

- chrono

6.471 std::greater< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::greater< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

6.471.1 Detailed Description

template<typename _Tp>
struct std::greater< _Tp >

One of the comparison functors.

6.471.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.472 std::ranges::greater Struct Reference

`#include <ranges_cmp.h>`

Public Types

- using **is_transparent**

Static Public Member Functions

- template<typename _Tp, typename _Up>
  static constexpr bool **operator()** (_Tp &&__t, _Up &&__u) noexcept(noexcept(std::declval< _Up >()< std::declval< _Tp >()))

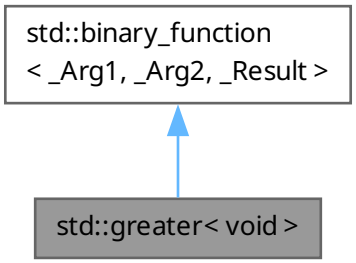
6.472.1 Detailed Description

ranges::greater function object type.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

6.473 std::greater< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::greater< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) > std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) > std::forward< _Up >(__u))
- template<typename _Tp, typename _Up>
  constexpr bool **operator()** (_Tp *__t, _Up *__u) const noexcept

6.473.1 Detailed Description

One of the comparison functors.

6.473.2 Member Typedef Documentation

first_argument_type

`typedef void std::binary_function< void, void, bool >::first_argument_type`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< void, void, bool >::result_type`
`result_type` is the return type

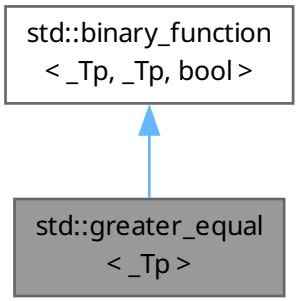
second_argument_type

`typedef void std::binary_function< void, void, bool >::second_argument_type`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.474 std::greater_equal< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::greater_equal< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

6.474.1 Detailed Description

template<typename _Tp>
struct std::greater_equal< _Tp >

One of the comparison functors.

6.474.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.475 std::ranges::greater_equal Struct Reference

`#include <ranges_cmp.h>`

Public Types

- using **is_transparent**

Static Public Member Functions

- template<typename _Tp, typename _Up>
  static constexpr bool **operator()** (_Tp &&__t, _Up &&__u) noexcept(noexcept(std::declval< _Tp >()< std::declval< _Up >()))

6.475.1 Detailed Description

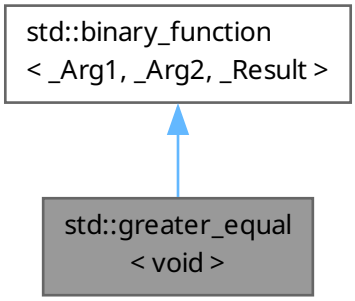
ranges::greater_equal function object type.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

6.476 std::greater_equal< void > Struct Reference

`#include <stl_function.h>`

Inheritance diagram for std::greater_equal< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) >=std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) >=std::forward< _Up >(__u))
- template<typename _Tp, typename _Up>
  constexpr bool **operator()** (_Tp *__t, _Up *__u) const noexcept

6.476.1 Detailed Description

One of the comparison functors.

6.476.2 Member Typedef Documentation

first_argument_type

```
typedef void std::binary_function< void, void, bool >::first_argument_type
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< void, void, bool >::result_type
```

`result_type` is the return type

second_argument_type

```
typedef void std::binary_function< void, void, bool >::second_argument_type
```

`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- stl_function.h

6.477 __gnu_cxx::random_condition::group_adjustor Struct Reference

`#include <throw_allocator.h>`

Public Member Functions

- **group_adjustor** (size_t size)

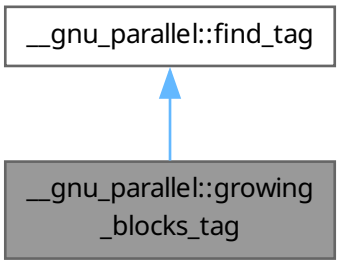
6.477.1 Detailed Description

Group condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.478 __gnu_parallel::growing_blocks_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::growing_blocks_tag:



6.478.1 Detailed Description

Selects the growing block size variant for std::find().

See also

_GLIBCXX_FIND_GROWING_BLOCKS

The documentation for this struct was generated from the following file:

- tags.h

6.479 std::gslice Class Reference

`#include <gslice.h>`

Public Member Functions

- gslice ()
- gslice (const gslice &)
- gslice (size_t __o, const valarray< size_t > &__l, const valarray< size_t > &__s)
- ~gslice ()
- gslice & operator= (const gslice &)
- valarray< size_t > size () const
- size_t start () const
- valarray< size_t > stride () const

Friends

- template<typename _Tp>
  class **valarray**

6.479.1 Detailed Description

Class defining multi-dimensional subset of an array.
The slice class represents a multi-dimensional subset of an array, specified by three parameter sets: start offset, size array, and stride array. The start offset is the index of the first element of the array that is part of the subset. The size and stride array describe each dimension of the slice. Size is the number of elements in that dimension, and stride is the distance in the array between successive elements in that dimension. Each dimension's size and stride is taken to begin at an array element described by the previous dimension. The size array and stride array must be the same size. For example, if you have offset==3, stride[0]==11, size[1]==3, stride[1]==3, then slice[0,0]==array[3], slice[0,1]==array[6], slice[0,2]==array[9], slice[1,0]==array[14], slice[1,1]==array[17], slice[1,2]==array[20].
The documentation for this class was generated from the following file:

- gslice.h

6.480 std::gslice_array< _Tp > Class Template Reference

`#include <gslice_array.h>`

Public Types

- typedef _Tp **value_type**

Public Member Functions

- gslice_array (const gslice_array &)
- template<class _Dom>
  void **operator%=** (const _Expr< _Dom, _Tp > &) const
- void operator%= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator&=** (const _Expr< _Dom, _Tp > &) const
- void operator&= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator*=** (const _Expr< _Dom, _Tp > &) const
- void operator*= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator+=** (const _Expr< _Dom, _Tp > &) const
- void operator+= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator-=** (const _Expr< _Dom, _Tp > &) const

- void operator-= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator/=** (const _Expr< _Dom, _Tp > &) const
- void operator/= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator<<=** (const _Expr< _Dom, _Tp > &) const
- void operator<<= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator=** (const _Expr< _Dom, _Tp > &) const
- void operator= (const _Tp &) const
- gslice_array & operator= (const gslice_array &)
- void operator= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator>>=** (const _Expr< _Dom, _Tp > &) const
- void operator>>= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator^=** (const _Expr< _Dom, _Tp > &) const
- void operator^= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator|=** (const _Expr< _Dom, _Tp > &) const
- void operator|= (const valarray< _Tp > &) const

Friends

- class **valarray< _Tp >**

6.480.1 Detailed Description

template<typename _Tp>
class std::gslice_array< _Tp >

Reference to multi-dimensional subset of an array.
A gslice_array is a reference to the actual elements of an array specified by a gslice. The way to get a gslice_array is to call operator[](gslice) on a valarray. The returned gslice_array then permits carrying operations out on the referenced subset of elements in the original valarray. For example, operator+=(valarray) will add values to the subset of elements in the underlying valarray this gslice_array refers to.

Parameters

Tp	Element type.

6.480.2 Member Function Documentation

operator%=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator%= (
            const valarray< _Tp > & ) const
```

Modulo slice elements by corresponding elements of *v*.

operator&=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator&= (
            const valarray< _Tp > & ) const
```

Logical and slice elements with corresponding elements of *v*.

operator*=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator*= (
            const valarray< _Tp > & ) const
```

Multiply slice elements by corresponding elements of *v*.

operator+=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator+= (
            const valarray< _Tp > & ) const
```

Add corresponding elements of *v* to slice elements.

operator-=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator-= (
            const valarray< _Tp > & ) const
```

Subtract corresponding elements of *v* from slice elements.

operator/=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator/= (
            const valarray< _Tp > & ) const
```

Divide slice elements by corresponding elements of *v*.

operator<<=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator<<= (
            const valarray< _Tp > & ) const
```

Left shift slice elements by corresponding elements of *v*.

operator>>=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator>>= (
            const valarray< _Tp > & ) const
```

Right shift slice elements by corresponding elements of *v*.

operator^=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator^= (
            const valarray< _Tp > & ) const
```

Logical xor slice elements with corresponding elements of *v*.

operator"" |=()

```
template<typename _Tp>
void std::gslice_array< _Tp >::operator|= (
            const valarray< _Tp > & ) const
```

Logical or slice elements with corresponding elements of *v*.
The documentation for this class was generated from the following files:

- valarray
- gslice_array.h

6.481 std::has_virtual_destructor< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.481.1 Detailed Description

template<typename _Tp>
struct std::has_virtual_destructor< _Tp >

has_virtual_destructor
The documentation for this struct was generated from the following file:

- type_traits

6.482 std::hash< _Tp > Struct Template Reference

```
#include <functional_hash.h>
```

6.482.1 Detailed Description

template<typename _Tp>
struct std::hash< _Tp >

Primary class template hash.
Primary class template hash, usable for enum types only.
The documentation for this struct was generated from the following file:

- string_view

6.483 std::hash< __debug::bitset< _Nb > > Struct Template Reference

```
#include <bitset>
```

Public Member Functions

- size_t **operator()** (const __debug::bitset< _Nb > &__b) const noexcept

6.483.1 Detailed Description

template<size_t _Nb>
struct std::hash< __debug::bitset< _Nb > >

std::hash specialization for bitset.
The documentation for this struct was generated from the following file:

- debug/bitset

6.484 std::hash< __debug::vector< bool, _Alloc > > Struct Template Reference

`#include <vector>`

Public Member Functions

- size_t **operator()** (const __debug::vector< bool, _Alloc > &__b) const noexcept

6.484.1 Detailed Description

template<typename _Alloc>
struct std::hash< __debug::vector< bool, _Alloc > >

std::hash specialization for vector<bool>.
The documentation for this struct was generated from the following file:

- debug/vector

6.485 std::hash< __gnu_cxx::__u16vstring > Struct Reference

`#include <vstring.h>`

Public Member Functions

- size_t **operator()** (const __gnu_cxx::__u16vstring &__s) const noexcept

6.485.1 Detailed Description

std::hash specialization for __u16vstring.
The documentation for this struct was generated from the following file:

- vstring.h

6.486 std::hash< __gnu_cxx::__u32vstring > Struct Reference

`#include <vstring.h>`

Public Member Functions

- size_t **operator()** (const __gnu_cxx::__u32vstring &__s) const noexcept

6.486.1 Detailed Description

std::hash specialization for __u32vstring.
The documentation for this struct was generated from the following file:

- vstring.h

6.487 std::hash< __gnu_cxx::__vstring > Struct Reference

`#include <vstring.h>`

Public Member Functions

- size_t **operator()** (const __gnu_cxx::__vstring &__s) const noexcept

6.487.1 Detailed Description

std::hash specialization for __vstring.
The documentation for this struct was generated from the following file:

- vstring.h

6.488 std::hash< __gnu_cxx::__wvstring > Struct Reference

`#include <vstring.h>`

Public Member Functions

- size_t **operator()** (const __gnu_cxx::__wvstring &__s) const noexcept

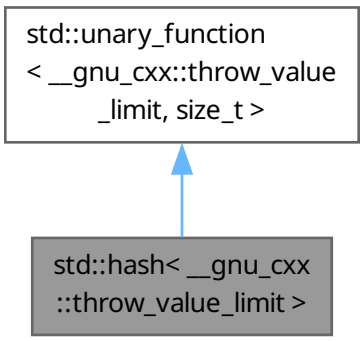
6.488.1 Detailed Description

std::hash specialization for __wvstring.
The documentation for this struct was generated from the following file:

- vstring.h

6.489 std::hash< __gnu_cxx::throw_value_limit > Struct Reference

`#include <throw_allocator.h>`
Inheritance diagram for std::hash< __gnu_cxx::throw_value_limit >:



Public Types

- typedef __gnu_cxx::throw_value_limit argument_type
- typedef size_t result_type

Public Member Functions

- size_t **operator()** (const __gnu_cxx::throw_value_limit &__val) const

6.489.1 Detailed Description

Explicit specialization of std::hash for __gnu_cxx::throw_value_limit.

6.489.2 Member Typedef Documentation

argument_type

```
typedef __gnu_cxx::throw_value_limit std::unary_function< __gnu_cxx::throw_value_limit, size_↩
t >::argument_type [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef size_t std::unary_function< __gnu_cxx::throw_value_limit, size_t >::result_type [inherited]
```

`result_type` is the return type

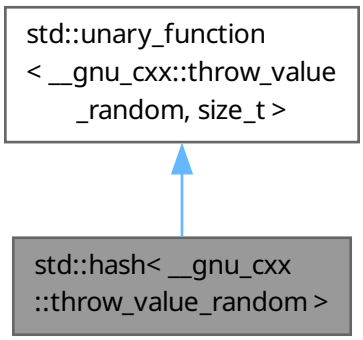
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.490 std::hash< __gnu_cxx::throw_value_random > Struct Reference

`#include <throw_allocator.h>`

Inheritance diagram for std::hash< __gnu_cxx::throw_value_random >:



Public Types

- typedef __gnu_cxx::throw_value_random argument_type
- typedef size_t result_type

Public Member Functions

- size_t **operator()** (const __gnu_cxx::throw_value_random &__val) const

6.490.1 Detailed Description

Explicit specialization of std::hash for __gnu_cxx::throw_value_random.

6.490.2 Member Typedef Documentation

argument_type

```
typedef __gnu_cxx::throw_value_random std::unary_function< __gnu_cxx::throw_value_random, size_t
>::argument_type [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef size_t std::unary_function< __gnu_cxx::throw_value_random, size_t >::result_type [inherited]
```

`result_type` is the return type

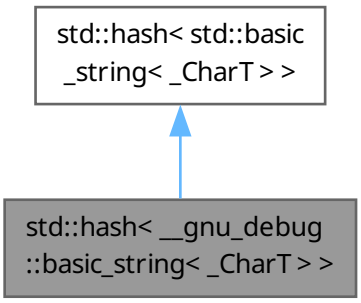
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.491 std::hash< __gnu_debug::basic_string< _CharT > > Struct Template Reference

`#include <string>`

Inheritance diagram for std::hash< __gnu_debug::basic_string< _CharT > >:



6.491.1 Detailed Description

template<typename _CharT>
struct std::hash< __gnu_debug::basic_string< _CharT > >

std::hash specialization for __gnu_debug::basic_string.

The documentation for this struct was generated from the following file:

- debug/string

6.492 std::hash< __shared_ptr< _Tp, _Lp > > Struct Template Reference

`#include <shared_ptr_base.h>`

Public Member Functions

- size_t **operator()** (const __shared_ptr< _Tp, _Lp > &__s) const noexcept

6.492.1 Detailed Description

template<typename _Tp, _Lock_policy _Lp>
struct std::hash< __shared_ptr< _Tp, _Lp > >

std::hash specialization for __shared_ptr.
The documentation for this struct was generated from the following file:

- shared_ptr_base.h

6.493 std::hash< _Tp * > Struct Template Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (_Tp *__p) const noexcept

6.493.1 Detailed Description

template<typename _Tp>
struct std::hash< _Tp * >

Partial specializations for pointer types.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.494 std::hash< basic_string< char, char_traits< char >, _Alloc > > Struct Template Reference

`#include <basic_string.h>`

Public Member Functions

- size_t **operator()** (const _StrT &__s) const noexcept

6.494.1 Detailed Description

template<typename _Alloc>
struct std::hash< basic_string< char, char_traits< char >, _Alloc > >

std::hash specialization for string.
The documentation for this struct was generated from the following file:

- basic_string.h

6.495 std::hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > > Struct Template Reference

`#include <basic_string.h>`

Public Member Functions

- size_t **operator()** (const _StrT &__s) const noexcept

6.495.1 Detailed Description

template<**typename _Alloc**>
struct std::hash< **basic_string**< **char16_t, char_traits**< **char16_t** >, **_Alloc** > >

std::hash specialization for u16string.
The documentation for this struct was generated from the following file:

- basic_string.h

6.496 std::hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > > Struct Template Reference

`#include <basic_string.h>`

Public Member Functions

- size_t **operator()** (const _StrT &__s) const noexcept

6.496.1 Detailed Description

template<**typename _Alloc**>
struct std::hash< **basic_string**< **char32_t, char_traits**< **char32_t** >, **_Alloc** > >

std::hash specialization for u32string.
The documentation for this struct was generated from the following file:

- basic_string.h

6.497 std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > > Struct Template Reference

`#include <basic_string.h>`

Public Member Functions

- size_t **operator()** (const _StrT &__s) const noexcept

6.497.1 Detailed Description

template<**typename _Alloc**>
struct std::hash< **basic_string**< **wchar_t, char_traits**< **wchar_t** >, **_Alloc** > >

std::hash specialization for wstring.
The documentation for this struct was generated from the following file:

- basic_string.h

6.498 std::hash< bool > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (bool __val) const noexcept

6.498.1 Detailed Description

Explicit specialization for bool.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.499 std::hash< char > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (char __val) const noexcept

6.499.1 Detailed Description

Explicit specialization for char.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.500 std::hash< char16_t > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (char16_t __val) const noexcept

6.500.1 Detailed Description

Explicit specialization for char16_t.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.501 std::hash< char32_t > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (char32_t __val) const noexcept

6.501.1 Detailed Description

Explicit specialization for char32_t.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.502 std::hash< double > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (double __val) const noexcept

6.502.1 Detailed Description

Specialization for double.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.503 std::hash< error_code > Struct Reference

`#include <system_error>`

Public Member Functions

- size_t **operator()** (const error_code &__e) const noexcept

6.503.1 Detailed Description

std::hash specialization for error_code.
The documentation for this struct was generated from the following file:

- system_error

6.504 std::hash< error_condition > Struct Reference

`#include <system_error>`

Public Member Functions

- size_t **operator()** (const error_condition &__e) const noexcept

6.504.1 Detailed Description

std::hash specialization for error_condition.
The documentation for this struct was generated from the following file:

- system_error

6.505 std::hash< experimental::optional< _Tp > > Struct Template Reference

`#include <optional>`

Public Types

- using **argument_type**
- using **result_type**

Public Member Functions

- size_t **operator()** (const experimental::optional< _Tp > &__t) const noexcept(noexcept(hash< _Tp > {}(*__t)))

6.505.1 Detailed Description

template<typename _Tp>
struct std::hash< experimental::optional< _Tp > >

std::hash partial specialization for experimental::optional
The documentation for this struct was generated from the following file:

- experimental/optional

6.506 std::hash< experimental::shared_ptr< _Tp > > Struct Template Reference

`#include <shared_ptr.h>`

Public Member Functions

- size_t **operator()** (const experimental::shared_ptr< _Tp > &__s) const noexcept

6.506.1 Detailed Description

template<typename _Tp>
struct std::hash< experimental::shared_ptr< _Tp > >

std::hash specialization for shared_ptr.
The documentation for this struct was generated from the following file:

- experimental/bits/shared_ptr.h

6.507 std::hash< float > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (float __val) const noexcept

6.507.1 Detailed Description

Specialization for float.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.508 std::hash< int > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (int __val) const noexcept

6.508.1 Detailed Description

Explicit specialization for int.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.509 std::hash< long > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (long __val) const noexcept

6.509.1 Detailed Description

Explicit specialization for long.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.510 std::hash< long double > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (long double __val) const noexcept

6.510.1 Detailed Description

Specialization for long double.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.511 std::hash< long long > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (long long __val) const noexcept

6.511.1 Detailed Description

Explicit specialization for long long.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.512 std::hash< shared_ptr< _Tp > > Struct Template Reference

`#include <shared_ptr.h>`

Public Member Functions

- size_t **operator()** (const shared_ptr< _Tp > &__s) const noexcept

6.512.1 Detailed Description

template<typename _Tp>
struct std::hash< shared_ptr< _Tp > >

std::hash specialization for shared_ptr.
The documentation for this struct was generated from the following file:

- bits/shared_ptr.h

6.513 std::hash< short > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (short __val) const noexcept

6.513.1 Detailed Description

Explicit specialization for short.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.514 std::hash< signed char > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (signed char __val) const noexcept

6.514.1 Detailed Description

Explicit specialization for signed char.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.515 std::hash< thread::id > Struct Reference

`#include <std_thread.h>`

Public Member Functions

- size_t **operator()** (const thread::id &__id) const noexcept

6.515.1 Detailed Description

std::hash specialization for thread::id.
The documentation for this struct was generated from the following file:

- std_thread.h

6.516 std::hash< type_index > Struct Reference

`#include <typeindex>`

Public Types

- typedef type_index **argument_type**
- typedef size_t **result_type**

Public Member Functions

- size_t **operator()** (const type_index &__ti) const noexcept

6.516.1 Detailed Description

std::hash specialization for type_index.
The documentation for this struct was generated from the following file:

- typeindex

6.517 std::hash< unique_ptr< _Tp, _Dp > > Struct Template Reference

`#include <unique_ptr.h>`

6.517.1 Detailed Description

template<typename _Tp, typename _Dp>
struct std::hash< unique_ptr< _Tp, _Dp > >

std::hash specialization for unique_ptr.
The documentation for this struct was generated from the following file:

- unique_ptr.h

6.518 std::hash< unsigned char > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (unsigned char __val) const noexcept

6.518.1 Detailed Description

Explicit specialization for unsigned char.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.519 std::hash< unsigned int > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (unsigned int __val) const noexcept

6.519.1 Detailed Description

Explicit specialization for unsigned int.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.520 std::hash< unsigned long > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (unsigned long __val) const noexcept

6.520.1 Detailed Description

Explicit specialization for unsigned long.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.521 std::hash< unsigned long long > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (unsigned long long __val) const noexcept

6.521.1 Detailed Description

Explicit specialization for unsigned long long.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.522 std::hash< unsigned short > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (unsigned short __val) const noexcept

6.522.1 Detailed Description

Explicit specialization for unsigned short.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.523 std::hash< wchar_t > Struct Reference

`#include <functional_hash.h>`

Public Member Functions

- size_t **operator()** (wchar_t __val) const noexcept

6.523.1 Detailed Description

Explicit specialization for wchar_t.
The documentation for this struct was generated from the following file:

- functional_hash.h

6.524 std::hash<::bitset< _Nb > > Struct Template Reference

`#include <bitset>`

Public Member Functions

- size_t **operator()** (const ::bitset< _Nb > &__b) const noexcept

6.524.1 Detailed Description

template<size_t _Nb>
struct std::hash<::bitset< _Nb > >

std::hash specialization for bitset.
The documentation for this struct was generated from the following file:

- bitset

6.525 std::hash<::vector< bool, _Alloc > > Struct Template Reference

`#include <stl_bvector.h>`

Public Member Functions

- size_t **operator()** (const ::vector< bool, _Alloc > &) const noexcept

6.525.1 Detailed Description

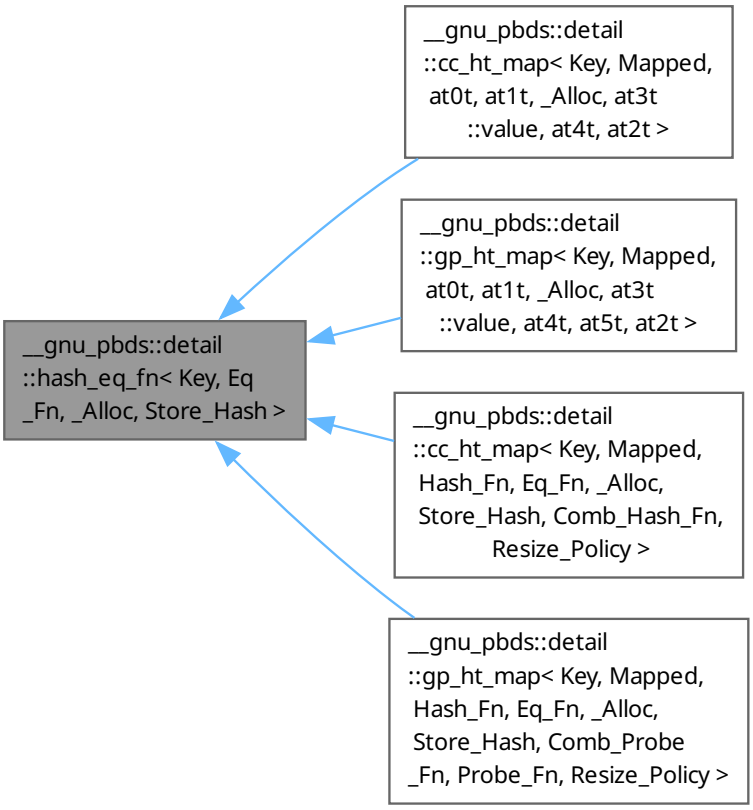
template<typename _Alloc>
struct std::hash<::vector< bool, _Alloc > >

std::hash specialization for vector<bool>.
The documentation for this struct was generated from the following files:

- stl_bvector.h
- vector.tcc

6.526 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash > Struct Template Reference

Inheritance diagram for __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >:



6.526.1 Detailed Description

template<typename Key, typename Eq_Fn, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >

Primary template.
The documentation for this struct was generated from the following file:

- hash_eq_fn.hpp

6.527 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false > Struct Template Reference

```
#include <hash_eq_fn.hpp>
```

Public Types

- typedef Eq_Fn **eq_fn_base**

- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**

Public Member Functions

- **hash_eq_fn** (const Eq_Fn &r_eq_fn)
- bool **operator()** (key_const_reference r_lhs_key, key_const_reference r_rhs_key) const
- void **swap** (const hash_eq_fn &other)

6.527.1 Detailed Description

template<typename Key, typename Eq_Fn, typename _Alloc>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >

Specialization 1 - The client requests that hash values not be stored.
The documentation for this struct was generated from the following file:

- hash_eq_fn.hpp

6.528 __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true > Struct Template Reference

`#include <hash_eq_fn.hpp>`

Public Types

- typedef Eq_Fn **eq_fn_base**
- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Public Member Functions

- **hash_eq_fn** (const Eq_Fn &r_eq_fn)
- bool **operator()** (key_const_reference r_lhs_key, size_type lhs_hash, key_const_reference r_rhs_key, size_type rhs_hash) const
- void **swap** (const hash_eq_fn &other)

6.528.1 Detailed Description

template<typename Key, class Eq_Fn, class _Alloc>
struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >

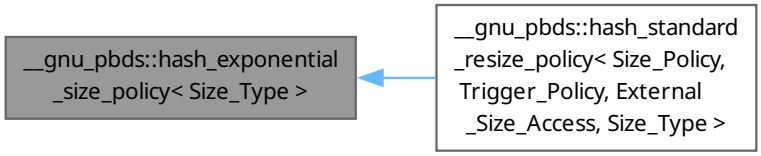
Specialization 2 - The client requests that hash values be stored.
The documentation for this struct was generated from the following file:

- hash_eq_fn.hpp

6.529 __gnu_pbds::hash_exponential_size_policy< Size_Type > Class Template Reference

`#include <hash_policy.hpp>`

Inheritance diagram for __gnu_pbds::hash_exponential_size_policy< Size_Type >:



Public Types

- typedef Size_Type **size_type**

Public Member Functions

- hash_exponential_size_policy (size_type start_size=8, size_type grow_factor=2)
- void **swap** (hash_exponential_size_policy< Size_Type > &other)

Protected Member Functions

- size_type **get_nearest_larger_size** (size_type size) const
- size_type **get_nearest_smaller_size** (size_type size) const

6.529.1 Detailed Description

template<typename Size_Type = std::size_t>
class __gnu_pbds::hash_exponential_size_policy< Size_Type >

A size policy whose sequence of sizes form an exponential sequence (typically powers of 2.

6.529.2 Constructor & Destructor Documentation

hash_exponential_size_policy()

```
template<typename Size_Type>
__gnu_pbds::hash_exponential_size_policy< Size_Type >::hash_exponential_size_policy (
            size_type start_size = 8,
            size_type grow_factor = 2)
```

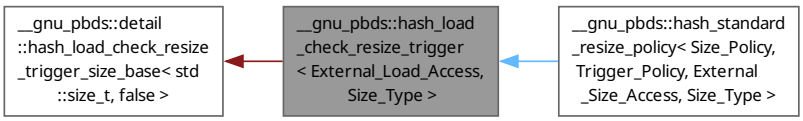
Default constructor, or onstructor taking a start_size, or constructor taking a start size and grow_factor. The policy will use the sequence of sizes start_size, start_size∗ grow_factor, start_size∗ grow_factor^2, ...
The documentation for this class was generated from the following file:

- hash_policy.hpp

6.530 __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type > Class Template Reference

```
#include <hash_policy.hpp>
```

Inheritance diagram for __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >:



Public Types

- enum { external_load_access }
- typedef Size_Type **size_type**

Public Member Functions

- hash_load_check_resize_trigger (float load_min=0.125, float load_max=0.5)
- std::pair< float, float > get_loads () const
- void set_loads (std::pair< float, float > load_pair)
- void **swap** (hash_load_check_resize_trigger &other)

Protected Member Functions

- bool **is_grow_needed** (size_type size, size_type num_entries) const
- bool **is_resize_needed** () const
- void notify_cleared ()
- void **notify_erase_search_collision** ()
- void **notify_erase_search_end** ()
- void **notify_erase_search_start** ()
- void **notify_erased** (size_type num_entries)
- void **notify_externally_resized** (size_type new_size)
- void **notify_find_search_collision** ()
- void **notify_find_search_end** ()
- void **notify_find_search_start** ()
- void **notify_insert_search_collision** ()
- void **notify_insert_search_end** ()
- void **notify_insert_search_start** ()
- void notify_inserted (size_type num_entries)
- void notify_resized (size_type new_size)

6.530.1 Detailed Description

template<bool External_Load_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >

A resize trigger policy based on a load check. It keeps the load factor between some load factors load_min and load↩
_max.

6.530.2 Member Enumeration Documentation

anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum
```

Enumerator

external_load_access	Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

6.530.3 Constructor & Destructor Documentation

hash_load_check_resize_trigger()

```
template<bool External_Load_Access, typename Size_Type>
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::hash_load_check_↩
resize_trigger (
            float load_min = 0.125,
            float load_max = 0.5)
```

Default constructor, or constructor taking load_min and load_max load factors between which this policy will keep the actual load.

6.530.4 Member Function Documentation

get_loads()

```
template<bool External_Load_Access, typename Size_Type>
std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size↩
_Type >::get_loads () const  [inline]
```

Returns a pair of the minimal and maximal loads, respectively.

notify_cleared()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↩
cleared ()  [protected]
```

Notifies the table was cleared.

notify_inserted()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↩
inserted (
            size_type num_entries)  [inline], [protected]
```

Notifies an element was inserted. The total number of entries in the table is num_entries.

notify_resized()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::notify_↩
resized (
            size_type new_size)  [protected]
```

Notifies the table was resized as a result of this object's signifying that a resize is needed.

References notify_resized().
Referenced by notify_resized().

set_loads()

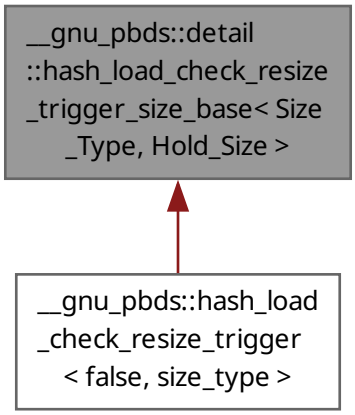
```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::set_loads (
            std::pair< float, float > load_pair)
```

Sets the loads through a pair of the minimal and maximal loads, respectively.
The documentation for this class was generated from the following file:

- hash_policy.hpp

6.531 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >:



6.531.1 Detailed Description

template<typename Size_Type, bool Hold_Size>
class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >

Primary template.
The documentation for this class was generated from the following file:

- hash_load_check_resize_trigger_size_base.hpp

6.532 __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true > Class Template Reference

```
#include <hash_load_check_resize_trigger_size_base.hpp>
```

Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- size_type **get_size** () const
- void **set_size** (size_type size)
- void **swap** (hash_load_check_resize_trigger_size_base &other)

6.532.1 Detailed Description

template<typename Size_Type>
class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >

Specializations.
The documentation for this class was generated from the following file:

- hash_load_check_resize_trigger_size_base.hpp

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

`#include <hash_map>`

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Ht::const_pointer **const_pointer**
- typedef _Ht::const_reference **const_reference**
- typedef _Tp **data_type**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Tp **mapped_type**
- typedef _Ht::pointer **pointer**
- typedef _Ht::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- template<class _InputIterator>
  hash_map (_InputIterator __f, _InputIterator __l)
- template<class _InputIterator>
  hash_map (_InputIterator __f, _InputIterator __l, size_type __n)
- template<class _InputIterator>
  hash_map (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)
- template<class _InputIterator>
  hash_map (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())

- **hash_map** (size_type __n)
- **hash_map** (size_type __n, const hasher &__hf)
- **hash_map** (size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator↩ _type())
- iterator **begin** ()
- const_iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key)
- pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator>
  void **insert** (_InputIterator __f, _InputIterator __l)
- pair< iterator, bool > **insert** (const value_type &__obj)
- pair< iterator, bool > **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- _Tp & **operator[ ]** (const key_type &__key)
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_map &__hs)

Friends

- template<class _K1, class _T1, class _HF, class _EqK, class _Al>
  bool **operator==** (const hash_map< _K1, _T1, _HF, _EqK, _Al > &, const hash_map< _K1, _T1, _HF, _EqK, _Al > &)

6.533.1 Detailed Description

template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>>
class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >

This is an SGI extension.

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

The documentation for this class was generated from the following file:

- hash_map

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

`#include <hash_map>`

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Ht::const_pointer **const_pointer**
- typedef _Ht::const_reference **const_reference**
- typedef _Tp **data_type**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Tp **mapped_type**
- typedef _Ht::pointer **pointer**
- typedef _Ht::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- template<class _InputIterator>
  hash_multimap (_InputIterator __f, _InputIterator __l)
- template<class _InputIterator>
  hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n)
- template<class _InputIterator>
  hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)
- template<class _InputIterator>
  hash_multimap (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &_↩
  _eql, const allocator_type &__a=allocator_type())
- **hash_multimap** (size_type __n)
- **hash_multimap** (size_type __n, const hasher &__hf)
- **hash_multimap** (size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__↩
  a=allocator_type())
- iterator **begin** ()
- const_iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key)
- pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)

- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator>
  void **insert** (_InputIterator __f, _InputIterator __l)
- iterator **insert** (const value_type &__obj)
- iterator **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_multimap &__hs)

Friends

- template<class _K1, class _T1, class _HF, class _EqK, class _Al>
  bool **operator==** (const hash_multimap< _K1, _T1, _HF, _EqK, _Al > &, const hash_multimap< _K1, _T1, _HF, _EqK, _Al > &)

6.534.1 Detailed Description

template<class _Key, class _Tp, class _HashFn = hash<_Key>, class _EqualKey = equal_to<_Key>, class _Alloc = allocator<_Tp>>
class __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >

This is an SGI extension.

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

The documentation for this class was generated from the following file:

- hash_map

6.535 __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference

`#include <hash_set>`

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Alloc::const_pointer **const_pointer**
- typedef _Alloc::const_reference **const_reference**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::const_iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Alloc::pointer **pointer**
- typedef _Alloc::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- template<class _InputIterator>
  hash_multiset (_InputIterator __f, _InputIterator __l)
- template<class _InputIterator>
  hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n)
- template<class _InputIterator>
  hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)
- template<class _InputIterator>
  hash_multiset (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- **hash_multiset** (size_type __n)
- **hash_multiset** (size_type __n, const hasher &__hf)
- **hash_multiset** (size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__↩ a=allocator_type())
- iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator>
  void **insert** (_InputIterator __f, _InputIterator __l)
- iterator **insert** (const value_type &__obj)
- iterator **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_multiset &hs)

Friends

- template<class _Val, class _HF, class _EqK, class _Al>
  bool **operator==** (const hash_multiset< _Val, _HF, _EqK, _Al > &, const hash_multiset< _Val, _HF, _EqK, _Al > &)

6.535.1 Detailed Description

template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>>
class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >

This is an SGI extension.

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

The documentation for this class was generated from the following file:

- hash_set

6.536 __gnu_pbds::hash_prime_size_policy Class Reference

`#include <hash_policy.hpp>`

Public Types

- typedef std::size_t size_type

Public Member Functions

- hash_prime_size_policy (size_type start_size=8)
- void **swap** (hash_prime_size_policy &other)

Protected Member Functions

- size_type **get_nearest_larger_size** (size_type size) const
- size_type **get_nearest_smaller_size** (size_type size) const

6.536.1 Detailed Description

A size policy whose sequence of sizes form a nearly-exponential sequence of primes.

6.536.2 Member Typedef Documentation

size_type

```
typedef std::size_t __gnu_pbds::hash_prime_size_policy::size_type
```

Size type.

6.536.3 Constructor & Destructor Documentation

hash_prime_size_policy()

```
__gnu_pbds::hash_prime_size_policy::hash_prime_size_policy (
            size_type start_size = 8)  [inline]
```

Default constructor, or onstructor taking a start_size The policy will use the sequence of sizes approximately start_size, start_size∗ 2, start_size∗ 2^2, ...

The documentation for this class was generated from the following file:

- hash_policy.hpp

6.537 __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc > Class Template Reference

`#include <hash_set>`

Public Types

- typedef _Ht::allocator_type **allocator_type**
- typedef _Ht::const_iterator **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef _Ht::difference_type **difference_type**
- typedef _Ht::hasher **hasher**
- typedef _Ht::const_iterator **iterator**
- typedef _Ht::key_equal **key_equal**
- typedef _Ht::key_type **key_type**
- typedef _Alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef _Ht::size_type **size_type**
- typedef _Ht::value_type **value_type**

Public Member Functions

- template<class _InputIterator>
  hash_set (_InputIterator __f, _InputIterator __l)
- template<class _InputIterator>
  hash_set (_InputIterator __f, _InputIterator __l, size_type __n)
- template<class _InputIterator>
  hash_set (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf)
- template<class _InputIterator>
  hash_set (_InputIterator __f, _InputIterator __l, size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_type())
- **hash_set** (size_type __n)
- **hash_set** (size_type __n, const hasher &__hf)
- **hash_set** (size_type __n, const hasher &__hf, const key_equal &__eql, const allocator_type &__a=allocator_↩ type())
- iterator **begin** () const
- size_type **bucket_count** () const
- void **clear** ()
- size_type **count** (const key_type &__key) const
- size_type **elems_in_bucket** (size_type __n) const
- bool **empty** () const
- iterator **end** () const
- pair< iterator, iterator > **equal_range** (const key_type &__key) const
- size_type **erase** (const key_type &__key)
- void **erase** (iterator __f, iterator __l)
- void **erase** (iterator __it)
- iterator **find** (const key_type &__key) const
- allocator_type **get_allocator** () const
- hasher **hash_funct** () const
- template<class _InputIterator>
  void **insert** (_InputIterator __f, _InputIterator __l)
- pair< iterator, bool > **insert** (const value_type &__obj)
- pair< iterator, bool > **insert_noresize** (const value_type &__obj)
- key_equal **key_eq** () const
- size_type **max_bucket_count** () const
- size_type **max_size** () const
- void **resize** (size_type __hint)
- size_type **size** () const
- void **swap** (hash_set &__hs)

Friends

- template<class _Val, class _HF, class _EqK, class _Al>
  bool **operator==** (const hash_set< _Val, _HF, _EqK, _Al > &, const hash_set< _Val, _HF, _EqK, _Al > &)

6.537.1 Detailed Description

template<class _Value, class _HashFcn = hash<_Value>, class _EqualKey = equal_to<_Value>, class _Alloc = allocator<_Value>>
class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >

This is an SGI extension.

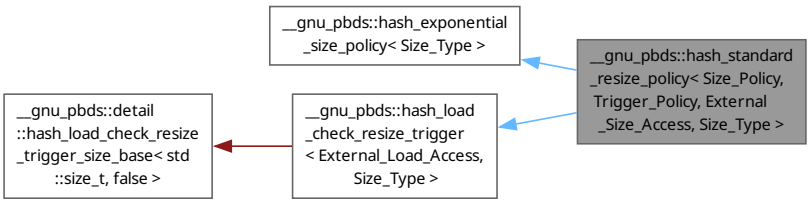
Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

The documentation for this class was generated from the following file:

- hash_set

6.538 __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > Class Template Reference

`#include <hash_policy.hpp>`
Inheritance diagram for __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >:



Public Types

- enum { external_load_access }
- enum { **external_size_access** }
- typedef Size_Policy **size_policy**
- typedef Size_Type **size_type**
- typedef Trigger_Policy **trigger_policy**

Public Member Functions

- hash_standard_resize_policy ()
- hash_standard_resize_policy (const Size_Policy &r_size_policy)
- hash_standard_resize_policy (const Size_Policy &r_size_policy, const Trigger_Policy &r_trigger_policy)
- size_type get_actual_size () const
- std::pair< float, float > get_loads () const
- Size_Policy & get_size_policy ()

- const Size_Policy & get_size_policy () const
- Trigger_Policy & get_trigger_policy ()
- const Trigger_Policy & get_trigger_policy () const
- void resize (size_type suggested_new_size)
- void set_loads (std::pair< float, float > load_pair)
- void **swap** (hash_exponential_size_policy< Size_Type > &other)
- void **swap** (hash_load_check_resize_trigger &other)
- void **swap** (hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type > &other)

Protected Member Functions

- size_type **get_nearest_larger_size** (size_type size) const
- size_type **get_nearest_smaller_size** (size_type size) const
- size_type get_new_size (size_type size, size_type num_used_e) const
- bool **is_grow_needed** (size_type size, size_type num_entries) const
- bool **is_resize_needed** () const
- void **notify_cleared** ()
- void **notify_erase_search_collision** ()
- void **notify_erase_search_end** ()
- void **notify_erase_search_start** ()
- void **notify_erased** (size_type num_e)
- void **notify_externally_resized** (size_type new_size)
- void **notify_find_search_collision** ()
- void **notify_find_search_end** ()
- void **notify_find_search_start** ()
- void **notify_insert_search_collision** ()
- void **notify_insert_search_end** ()
- void **notify_insert_search_start** ()
- void **notify_inserted** (size_type num_e)
- void **notify_resized** (size_type new_size)

6.538.1 Detailed Description

template<typename Size_Policy = hash_exponential_size_policy<>, typename Trigger_Policy = hash_load_check_resize_trigger<>, bool External_Size_Access = false, typename Size_Type = std::size_t>
class __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >

A resize policy which delegates operations to size and trigger policies.

6.538.2 Member Enumeration Documentation

anonymous enum

```
template<bool External_Load_Access = false, typename Size_Type = std::size_t>
anonymous enum  [inherited]
```

Enumerator

external_load_access	Specifies whether the load factor can be accessed externally. The two options have different trade-offs in terms of flexibility, genericity, and encapsulation.

6.538.3 Constructor & Destructor Documentation

hash_standard_resize_policy() [1/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size↩
_Type >::hash_standard_resize_policy ()
```

Default constructor.

hash_standard_resize_policy() [2/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size↩
_Type >::hash_standard_resize_policy (
            const Size_Policy & r_size_policy)
```

constructor taking some policies r_size_policy will be copied by the Size_Policy object of this object.

hash_standard_resize_policy() [3/3]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size↩
_Type >::hash_standard_resize_policy (
            const Size_Policy & r_size_policy,
            const Trigger_Policy & r_trigger_policy)
```

constructor taking some policies. r_size_policy will be copied by the Size_Policy object of this object. r_trigger_policy will be copied by the Trigger_Policy object of this object.

6.538.4 Member Function Documentation

get_actual_size()

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::↩
size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_↩
Access, Size_Type >::get_actual_size () const  [inline]
```

Returns the actual size of the container.

get_loads()

```
template<bool External_Load_Access, typename Size_Type>
std::pair< float, float > __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size↩
_Type >::get_loads () const  [inline], [inherited]
```

Returns a pair of the minimal and maximal loads, respectively.

get_new_size()

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >::↩
size_type __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_↩
Access, Size_Type >::get_new_size (
```

```
            size_type size,
            size_type num_used_e) const  [protected]
```

Queries what the new size should be, when the container is resized naturally. The current __size of the container is size, and the number of used entries within the container is num_used_e.

get_size_policy() [1/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↩
Size_Access, Size_Type >::get_size_policy ()
```

Access to the Size_Policy object used.

get_size_policy() [2/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
const Size_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External↩
_Size_Access, Size_Type >::get_size_policy () const
```

Const access to the Size_Policy object used.

get_trigger_policy() [1/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_↩
Size_Access, Size_Type >::get_trigger_policy ()
```

Access to the Trigger_Policy object used.

get_trigger_policy() [2/2]

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
const Trigger_Policy & __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy,
External_Size_Access, Size_Type >::get_trigger_policy () const
```

Access to the Trigger_Policy object used.

resize()

```
template<typename Size_Policy, typename Trigger_Policy, bool External_Size_Access, typename Size↩
_Type>
void __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access,
Size_Type >::resize (
            size_type suggested_new_size)
```

Resizes the container to suggested_new_size, a suggested size (the actual size will be determined by the Size_Policy object).

set_loads()

```
template<bool External_Load_Access, typename Size_Type>
void __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >::set_loads (
            std::pair< float, float > load_pair)  [inherited]
```

Sets the loads through a pair of the minimal and maximal loads, respectively.
The documentation for this class was generated from the following file:

- hash_policy.hpp

6.539 std::chrono::hh_mm_ss< _Duration > Class Template Reference

`#include <chrono>`

Public Types

- using **precision**

Public Member Functions

- constexpr **hh_mm_ss** (_Duration __d)
- constexpr chrono::hours **hours** () const noexcept
- constexpr bool **is_negative** () const noexcept
- constexpr chrono::minutes **minutes** () const noexcept
- constexpr **operator precision** () const noexcept
- constexpr chrono::seconds **seconds** () const noexcept
- constexpr precision **subseconds** () const noexcept
- constexpr precision **to_duration** () const noexcept

Static Public Attributes

- static constexpr unsigned **fractional_width**

6.539.1 Detailed Description

template<typename _Duration>
class std::chrono::hh_mm_ss< _Duration >

Utility for splitting a duration into hours, minutes, and seconds
This is a convenience type that provides accessors for the constituent parts (hours, minutes, seconds and subseconds) of a duration.

Since

C++20

The documentation for this class was generated from the following file:

- chrono

6.540 std::locale::id Class Reference

`#include <locale_classes.h>`

Public Member Functions

- id ()
- size_t **_M_id** () const throw ()

Friends

- template<typename _Facet>
  const _Facet * **__try_use_facet** (const locale &__loc)
- template<typename _Facet>
  bool has_facet (const locale &__loc) throw ()
- class **locale**
- class **locale::_Impl**
- template<typename _Facet>
  const _Facet & use_facet (const locale &__loc)

6.540.1 Detailed Description

Facet ID class.
The ID class provides facets with an index used to identify them. Every facet class must define a public static member locale::id, or be derived from a facet that provides this member, otherwise the facet cannot be used in a locale. The locale::id ensures that each class type gets a unique identifier.

6.540.2 Constructor & Destructor Documentation

id()

```
std::locale::id::id ()  [inline]
```
Constructor.

6.540.3 Friends And Related Symbol Documentation

has_facet

```
template<typename _Facet>
bool has_facet (
            const locale & __loc) throw ( )   [friend]
```
Test for the presence of a facet.
has_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Template Parameters

_Facet	The facet type to test the presence of.

Parameters

__loc	The locale to test.

Returns

true if __loc contains a facet of type _Facet, else false.

use_facet

```
template<typename _Facet>
const _Facet & use_facet (
            const locale & __loc)  [friend]
```
Return a facet.
use_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has_↩ facet(locale) is true, there is a suitable facet to return. It throws std::bad_cast if the locale doesn't contain a facet of type Facet.

Template Parameters

_Facet	The facet type to access.

Parameters

__loc	The locale to use.

Returns

Reference to facet of type Facet.

Exceptions

std::bad_cast	if `__loc` doesn't contain a facet of type _Facet.

The documentation for this class was generated from the following file:

- locale_classes.h

6.541 std::thread::id Class Reference

`#include <thread>`

Public Member Functions

- **id** (native_handle_type __id)

Friends

- struct **hash**< **id** >
- template<class _CharT, class _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__out, id __id)
- strong_ordering **operator**<=> (id __x, id __y) noexcept
- bool **operator==** (id __x, id __y) noexcept
- class **thread**

Related Symbols

(Note that these are not member symbols.)

- bool **operator==** (thread::id __x, thread::id __y) noexcept
- strong_ordering **operator**<=> (thread::id __x, thread::id __y) noexcept
- template<class _CharT, class _Traits>
  basic_ostream< _CharT, _Traits > & **operator**<< (basic_ostream< _CharT, _Traits > &__out, thread::id __id)

6.541.1 Detailed Description

A std::thread::id is a unique identifier for a thread.

Since

C++11

The documentation for this class was generated from the following files:

- std_thread.h
- thread

6.542 std::identity Struct Reference

`#include <ranges_cmp.h>`

Public Types

- using **is_transparent**

Static Public Member Functions

- template<typename _Tp>
  static constexpr _Tp && **operator()** (_Tp &&__t) noexcept

6.542.1 Detailed Description

[func.identity] The identity function.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

6.543 std::experimental::fundamentals_v1::in_place_t Struct Reference

`#include <optional>`

6.543.1 Detailed Description

Tag type for in-place construction.
The documentation for this struct was generated from the following file:

- experimental/optional

6.544 std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > Class Template Reference

`#include <random>`

Public Types

- typedef _UIntType result_type

Public Member Functions

- independent_bits_engine ()
- independent_bits_engine (_RandomNumberEngine &&__rng)
- template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
  independent_bits_engine (_Sseq &__q)
- independent_bits_engine (const _RandomNumberEngine &__rng)
- independent_bits_engine (result_type __s)
- const _RandomNumberEngine & base () const noexcept
- void discard (unsigned long long __z)
- result_type operator() ()
- void seed ()
- template<typename _Sseq>
  _If_seed_seq< _Sseq > seed (_Sseq &__q)
- void seed (result_type __s)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Friends

- bool operator== (const independent_bits_engine &__lhs, const independent_bits_engine &__rhs)
- template<typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)

6.544.1 Detailed Description

template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >

Produces random numbers by combining random numbers from some base engine to produce random numbers with a specified number of bits __w.

Since

C++11

6.544.2 Member Typedef Documentation

result_type

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
typedef _UIntType std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::result_↩
type
```

The type of the generated random value.

6.544.3 Constructor & Destructor Documentation

independent_bits_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine ()
[inline]
```

Constructs a default independent_bits_engine engine.
The underlying engine is default constructed as well.
Referenced by operator==.

independent_bits_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
            const _RandomNumberEngine & __rng)  [inline], [explicit]
```

Copy constructs a independent_bits_engine engine.
Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

independent_bits_engine() [3/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
            _RandomNumberEngine && __rng)  [inline], [explicit]
```

Move constructs a independent_bits_engine engine.
Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

References std::move().

independent_bits_engine() [4/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
            result_type __s)  [inline], [explicit]
```

Seed constructs a independent_bits_engine engine.
Constructs the underlying generator engine seeded with __s.

Parameters

_↩ _s	A seed value for the base class engine.

independent_bits_engine() [5/5]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::independent_bits_engine (
            _Sseq & __q)  [inline], [explicit]
```

Generator construct a independent_bits_engine engine.

Parameters

_↩ _q	A seed sequence.

6.544.4 Member Function Documentation

base()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
const _RandomNumberEngine & std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType
>::base () const  [inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

discard()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
```

```
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::discard (
            unsigned long long __z)  [inline]
```
Discard a sequence of random numbers.

max()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
constexpr result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::max ()
[inline], [static], [constexpr]
```
Gets the maximum value in the generated random number range.

min()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
constexpr result_type std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::min ()
[inline], [static], [constexpr]
```
Gets the minimum value in the generated random number range.

operator()()

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::result_type std::independent_bits_engine<
_RandomNumberEngine, __w, _UIntType >::operator() ()
```
Gets the next value in the generated random number sequence.
References std::__lg(), std::__numeric_limits_base::digits, and std::numeric_limits< _Tp >::max().

seed() [1/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed ()  [inline]
```
Reseeds the independent_bits_engine object with the default seed for the underlying base class generator engine.

seed() [2/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed
(
            _Sseq & __q)  [inline]
```
Reseeds the independent_bits_engine object with the given seed sequence.

Parameters

_↩ _q	A seed generator function.

seed() [3/3]

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
void std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::seed (
            result_type __s)  [inline]
```
Reseeds the independent_bits_engine object with the default seed for the underlying base class generator engine.

6.544.5 Friends And Related Symbol Documentation

operator==

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
bool operator== (
             const independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __lhs,
             const independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __rhs)  [friend]
```

Compares two independent_bits_engine random number generator objects of the same type for equality.

Parameters

__lhs	A independent_bits_engine random number generator object.
__rhs	Another independent_bits_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

References independent_bits_engine().

operator>>

```
template<typename _RandomNumberEngine, size_t __w, typename _UIntType>
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
             std::basic_istream< _CharT, _Traits > & __is,
             std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > & __x)  [friend]
```

Extracts the current state of a % subtract_with_carry_engine random number generator engine __x from the input stream __is.

Parameters

__is	An input stream.
__x	A independent_bits_engine random number generator engine.

Returns

The input stream with the state of __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.545 std::indirect_array< _Tp > Class Template Reference

```
#include <indirect_array.h>
```

Public Types

- typedef _Tp **value_type**

Public Member Functions

- indirect_array (const indirect_array &)
- template<class _Dom>
  void **operator%=** (const _Expr< _Dom, _Tp > &) const
- void operator%= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator&=** (const _Expr< _Dom, _Tp > &) const
- void operator&= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator*=** (const _Expr< _Dom, _Tp > &) const
- void operator*= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator+=** (const _Expr< _Dom, _Tp > &) const
- void operator+= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator-=** (const _Expr< _Dom, _Tp > &) const
- void operator-= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator/=** (const _Expr< _Dom, _Tp > &) const
- void operator/= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator<<=** (const _Expr< _Dom, _Tp > &) const
- void operator<<= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator=** (const _Expr< _Dom, _Tp > &) const
- void operator= (const _Tp &) const
- indirect_array & operator= (const indirect_array &)
- void operator= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator>>=** (const _Expr< _Dom, _Tp > &) const
- void operator>>= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator^=** (const _Expr< _Dom, _Tp > &) const
- void operator^= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator|=** (const _Expr< _Dom, _Tp > &) const
- void operator|= (const valarray< _Tp > &) const

Friends

- class **gslice_array**< **_Tp** >
- class **valarray**< **_Tp** >

6.545.1 Detailed Description

template<**class _Tp**>
class std::indirect_array< **_Tp** >

Reference to arbitrary subset of an array.
An indirect_array is a reference to the actual elements of an array specified by an ordered array of indices. The way to get an indirect_array is to call operator[](valarray<size_t>) on a valarray. The returned indirect_array then permits carrying operations out on the referenced subset of elements in the original valarray.

For example, if an indirect_array is obtained using the array (4,2,0) as an argument, and then assigned to an array containing (1,2,3), then the underlying array will have array[0]==3, array[2]==2, and array[4]==1.

Parameters

Tp	Element type.

6.545.2 Member Function Documentation

operator%=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator%= (
            const valarray< _Tp > & ) const
```

Modulo slice elements by corresponding elements of *v*.

operator&=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator&= (
            const valarray< _Tp > & ) const
```

Logical and slice elements with corresponding elements of *v*.

operator*=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator*= (
            const valarray< _Tp > & ) const
```

Multiply slice elements by corresponding elements of *v*.

operator+=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator+= (
            const valarray< _Tp > & ) const
```

Add corresponding elements of *v* to slice elements.

operator-=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator-= (
            const valarray< _Tp > & ) const
```

Subtract corresponding elements of *v* from slice elements.

operator/=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator/= (
            const valarray< _Tp > & ) const
```

Divide slice elements by corresponding elements of *v*.

operator<<=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator<<= (
```

```
            const valarray< _Tp > & ) const
```

Left shift slice elements by corresponding elements of *v*.

operator>>=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator>>= (
            const valarray< _Tp > & ) const
```

Right shift slice elements by corresponding elements of *v*.

operator^=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator^= (
            const valarray< _Tp > & ) const
```

Logical xor slice elements with corresponding elements of *v*.

operator"|=()

```
template<class _Tp>
void std::indirect_array< _Tp >::operator|= (
            const valarray< _Tp > & ) const
```

Logical or slice elements with corresponding elements of *v*.
The documentation for this class was generated from the following files:

- valarray
- indirect_array.h

6.546 std::initializer_list< _E > Class Template Reference

`#include <initializer_list>`

Public Types

- typedef const _E * **const_iterator**
- typedef const _E & **const_reference**
- typedef const _E * **iterator**
- typedef const _E & **reference**
- typedef size_t **size_type**
- typedef _E **value_type**

Public Member Functions

- constexpr const_iterator **begin** () const noexcept
- constexpr const value_type * **data** () const noexcept
- constexpr bool **empty** () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr size_type **size** () const noexcept

6.546.1 Detailed Description

template<class _E>
class std::initializer_list< _E >

initializer_list
The documentation for this class was generated from the following file:

- initializer_list

6.547 std::inout_ptr_t< _Smart, _Pointer, _Args > Class Template Reference

`#include <>`

Public Member Functions

- **inout_ptr_t** (_Smart &__smart, _Args... __args)
- **inout_ptr_t** (const inout_ptr_t &)=delete
- **operator _Pointer** * () const noexcept
- **operator void** ** () const noexcept

6.547.1 Detailed Description

template<typename _Smart, typename _Pointer, typename... _Args>
class std::inout_ptr_t< _Smart, _Pointer, _Args >

Smart pointer adaptor for functions taking an inout pointer parameter.

Template Parameters

_Smart	The type of pointer to adapt.
_Pointer	The type of pointer to convert to.
_Args...	Argument types used when resetting the smart pointer.

Since

C++23

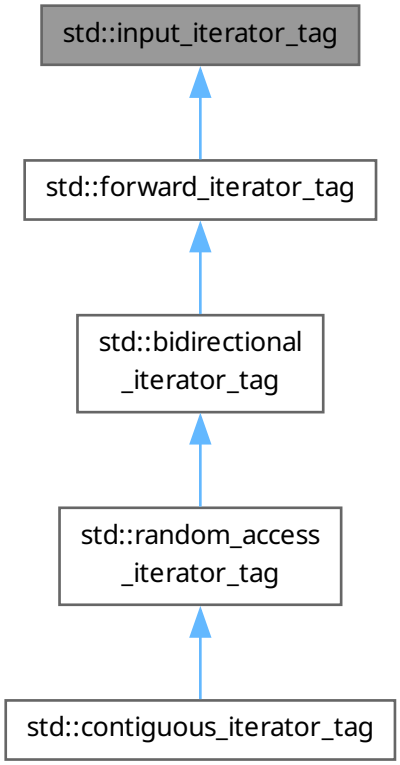
The documentation for this class was generated from the following file:

- out_ptr.h

6.548 std::input_iterator_tag Struct Reference

`#include <stl_iterator_base_types.h>`

Inheritance diagram for std::input_iterator_tag:



6.548.1 Detailed Description

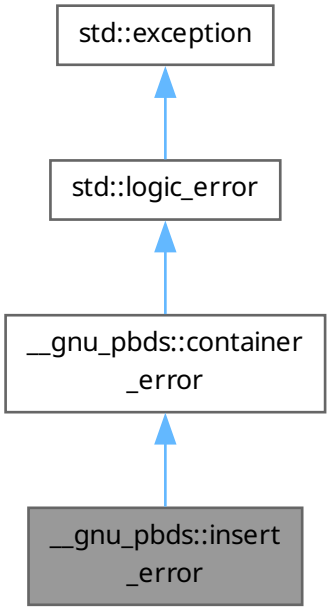
Marking input iterators.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

6.549 __gnu_pbds::insert_error Struct Reference

`#include <exception.hpp>`

Inheritance diagram for __gnu_pbds::insert_error:



Public Member Functions

- virtual const char * what () const noexcept

6.549.1 Detailed Description

An entry cannot be inserted into a container object for logical reasons (not, e.g., if memory is unabvailable, in which case the allocator_type's exception will be thrown).

6.549.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

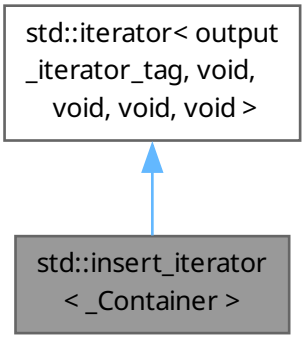
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
References what().
Referenced by what().
The documentation for this struct was generated from the following file:

- exception.hpp

6.550 std::insert_iterator< _Container > Class Template Reference

```
#include <stl_iterator.h>
```

Inheritance diagram for std::insert_iterator< _Container >:



Public Types

- typedef _Container container_type
- using **difference_type**
- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

Public Member Functions

- constexpr insert_iterator (_Container &__x, _Iter __i)
- constexpr insert_iterator & operator∗ ()
- constexpr insert_iterator & operator++ ()
- constexpr insert_iterator & operator++ (int)
- constexpr insert_iterator & operator= (const typename _Container::value_type &__value)
- constexpr insert_iterator & **operator=** (typename _Container::value_type &&__value)

Protected Attributes

- _Container ∗ **container**
- _Iter **iter**

6.550.1 Detailed Description

template<**typename _Container**>
class std::insert_iterator< **_Container** >

Turns assignment into insertion.
These are output iterators, constructed from a container-of-T. Assigning a T to the iterator inserts it in the container at the iterator's position, rather than overwriting the value at that position.
(Sequences will actually insert a *copy* of the value before the iterator's position.)
Tip: Using the inserter function to create these iterators can save typing.

6.550.2 Member Typedef Documentation

container_type

```
template<typename _Container>
typedef _Container std::insert_iterator< _Container >::container_type
```

A nested typedef for the type of whatever container you used.

iterator_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::↩
iterator_category  [inherited]
```

One of the tag types.

pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer  [inherited]
```

This type represents a pointer-to-value_type.

reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference  [inherited]
```

This type represents a reference-to-value_type.

value_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type  [inherited]
```

The type "pointed to" by the iterator.

6.550.3 Constructor & Destructor Documentation

insert_iterator()

```
template<typename _Container>
std::insert_iterator< _Container >::insert_iterator (
            _Container & __x,
            _Iter __i)  [inline], [constexpr]
```

The only way to create this iterator is with a container and an initial position (a normal iterator into the container).
References std::__addressof().

6.550.4 Member Function Documentation

operator∗()

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator* ()  [inline], [nodiscard], [constexpr]
```

Simply returns ∗this.

operator++() [1/2]

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator++ ()  [inline], [constexpr]
```

Simply returns ∗this. (This iterator does not *move*.).

operator++() [2/2]

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator++ (
            int )  [inline], [constexpr]
```

Simply returns *this. (This iterator does not *move*.).

operator=()

```
template<typename _Container>
insert_iterator & std::insert_iterator< _Container >::operator= (
            const typename _Container::value_type & __value)  [inline], [constexpr]
```

Parameters

__value	An instance of whatever type container_type::const_reference is; presumably a reference-to-const T for container<T>.

Returns

This iterator, for chained operations.

This kind of iterator maintains its own position in the container. Assigning a value to the iterator will insert the value into the container at the place before the iterator.
The position is maintained such that subsequent assignments will insert values immediately after one another. For example,

```
// vector v contains A and Z

insert_iterator i (v, ++v.begin());
i = 1;
i = 2;
i = 3;

// vector v contains A, 1, 2, 3, and Z
```

The documentation for this class was generated from the following file:

- bits/stl_iterator.h

6.551 std::integer_sequence< _Tp, _Idx > Struct Template Reference

`#include <utility.h>`

Public Types

- typedef _Tp **value_type**

Static Public Member Functions

- static constexpr size_t **size** () noexcept

6.551.1 Detailed Description

template<typename _Tp, _Tp... _Idx>
struct std::integer_sequence< _Tp, _Idx >

Class template integer_sequence.
The documentation for this struct was generated from the following file:

- utility.h

6.552 std::integral_constant< _Tp, __v > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr _Tp **value**

6.552.1 Detailed Description

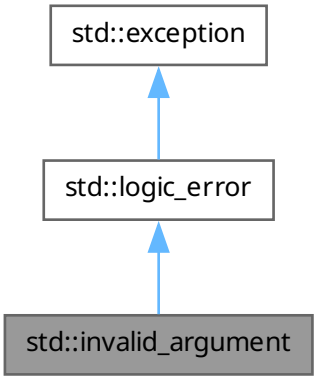
template<typename _Tp, _Tp __v>
struct std::integral_constant< _Tp, __v >

integral_constant
The documentation for this struct was generated from the following file:

- type_traits

6.553 std::invalid_argument Class Reference

`#include <stdexcept_except.h>`
Inheritance diagram for std::invalid_argument:



Public Member Functions

- **invalid_argument** (const char *)
- **invalid_argument** (const invalid_argument &)=default

- **invalid_argument** (const string &__arg)
- **invalid_argument** (invalid_argument &&)=default
- invalid_argument & **operator=** (const invalid_argument &)=default
- invalid_argument & **operator=** (invalid_argument &&)=default
- virtual const char * what () const noexcept

6.553.1 Detailed Description

Thrown to report invalid arguments to functions.

6.553.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

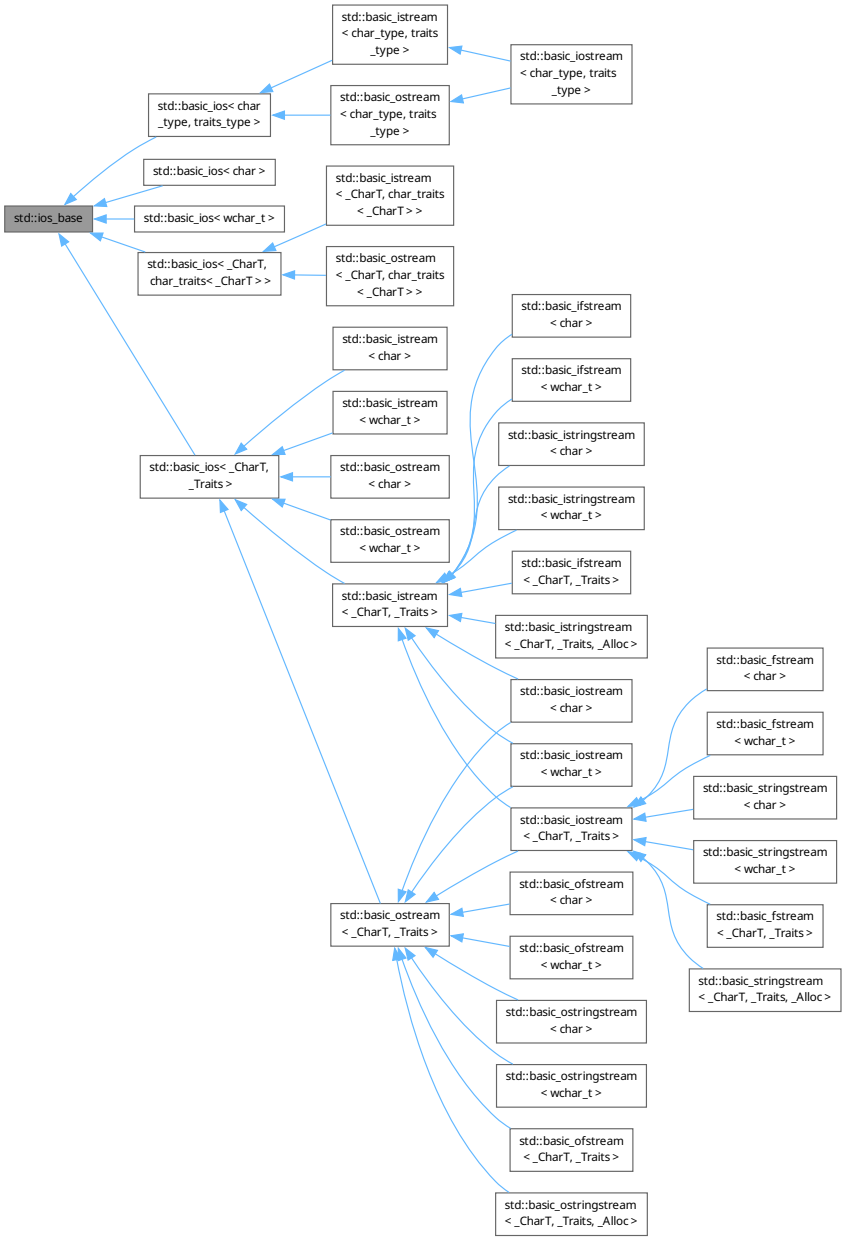
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
References what().
Referenced by what().
The documentation for this class was generated from the following file:

- stdexcept_except.h

6.554 std::ios_base Class Reference

```
#include <ios_base.h>
```

Inheritance diagram for std::ios_base:



Classes

- class failure

Public Types

- enum event { **erase_event** , **imbue_event** , **copyfmt_event** }
- typedef void(∗ event_callback) (event __e, ios_base &__b, int __i)
- typedef _Ios_Fmtflags fmtflags

- typedef _Ios_Iostate iostate
- typedef _Ios_Openmode openmode
- typedef _Ios_Seekdir seekdir

Public Member Functions

- **ios_base** (const ios_base &)=delete
- virtual ~ios_base ()
- const locale & _M_getloc () const
- fmtflags flags () const
- fmtflags flags (fmtflags __fmtfl)
- locale getloc () const
- locale imbue (const locale &__loc) throw ()
- long & iword (int __ix)
- ios_base & **operator=** (const ios_base &)=delete
- streamsize precision () const
- streamsize precision (streamsize __prec)
- void *& pword (int __ix)
- void register_callback (event_callback __fn, int __index)
- fmtflags setf (fmtflags __fmtfl)
- fmtflags setf (fmtflags __fmtfl, fmtflags __mask)
- void unsetf (fmtflags __mask)
- streamsize width () const
- streamsize width (streamsize __wide)

Static Public Member Functions

- static bool sync_with_stdio (bool __sync=true)
- static int xalloc () throw ()

Static Public Attributes

- static const openmode **__noreplace**
- static const fmtflags adjustfield
- static const openmode app
- static const openmode ate
- static const iostate badbit
- static const fmtflags basefield
- static const seekdir beg
- static const openmode binary
- static const fmtflags boolalpha
- static const seekdir cur
- static const fmtflags dec
- static const seekdir end
- static const iostate eofbit
- static const iostate failbit
- static const fmtflags fixed
- static const fmtflags floatfield
- static const iostate goodbit
- static const fmtflags hex
- static const openmode in
- static const fmtflags internal

- static const fmtflags left
- static const fmtflags oct
- static const openmode out
- static const fmtflags right
- static const fmtflags scientific
- static const fmtflags showbase
- static const fmtflags showpoint
- static const fmtflags showpos
- static const fmtflags skipws
- static const openmode trunc
- static const fmtflags unitbuf
- static const fmtflags uppercase

Protected Types

- enum { **_S_local_word_size** }

Protected Member Functions

- void **_M_call_callbacks** (event __ev) throw ()
- void **_M_dispose_callbacks** (void) throw ()
- _Words & **_M_grow_words** (int __index, bool __iword)
- void **_M_init** () throw ()
- void **_M_move** (ios_base &) noexcept
- void **_M_swap** (ios_base &__rhs) noexcept

Protected Attributes

- _Callback_list * **_M_callbacks**
- iostate **_M_exception**
- fmtflags **_M_flags**
- locale **_M_ios_locale**
- _Words **_M_local_word** [_S_local_word_size]
- streamsize **_M_precision**
- iostate **_M_streambuf_state**
- streamsize **_M_width**
- _Words * **_M_word**
- int **_M_word_size**
- _Words **_M_word_zero**

6.554.1 Detailed Description

The base of the I/O class hierarchy.
This class defines everything that can be defined about I/O that does not depend on the type of characters being input or output. Most people will only see `ios_base` when they need to specify the full name of the various I/O flags (e.g., the openmodes).

6.554.2 Member Typedef Documentation

event_callback

```
typedef void(* std::ios_base::event_callback) (event __e, ios_base &__b, int __i)
```

The type of an event callback function.

Parameters

__e	One of the members of the event enum.
__b	Reference to the ios_base object.
__i	The integer provided when the callback was registered.

Event callbacks are user defined functions that get called during several ios_base and basic_ios functions, specifically imbue(), copyfmt(), and ~ios().

fmtflags

```
typedef _Ios_Fmtflags std::ios_base::fmtflags
```

This is a bitmask type.

_Ios_Fmtflags is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type fmtflags are:

- boolalpha
- dec
- fixed
- hex
- internal
- left
- oct
- right
- scientific
- showbase
- showpoint
- showpos
- skipws
- unitbuf
- uppercase
- adjustfield

- basefield
- floatfield

iostate

`typedef _Ios_Iostate std::ios_base::iostate`
This is a bitmask type.
_Ios_Iostate is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type iostate are:

- badbit
- eofbit
- failbit
- goodbit

openmode

`typedef _Ios_Openmode std::ios_base::openmode`
This is a bitmask type.
_Ios_Openmode is implementation-defined, but it is valid to perform bitwise operations on these values and expect the Right Thing to happen. Defined objects of type openmode are:

- app
- ate
- binary
- in
- out
- trunc

seekdir

`typedef _Ios_Seekdir std::ios_base::seekdir`
This is an enumerated type.
_Ios_Seekdir is implementation-defined. Defined values of type seekdir are:

- beg
- cur, equivalent to `SEEK_CUR` in the C standard library.
- end, equivalent to `SEEK_END` in the C standard library.

6.554.3 Member Enumeration Documentation

event

`enum std::ios_base::event`
The set of events that may be passed to an event callback.
erase_event is used during ~ios() and copyfmt(). imbue_event is used during imbue(). copyfmt_event is used during copyfmt().

6.554.4 Constructor & Destructor Documentation

~ios_base()

```
virtual std::ios_base::~ios_base ()  [virtual]
```

Invokes each callback with erase_event. Destroys local storage.
Note that the ios_base object for the standard streams never gets destroyed. As a result, any callbacks registered with the standard streams will not get invoked with erase_event (unless copyfmt is used).

6.554.5 Member Function Documentation

_M_getloc()

```
const locale & std::ios_base::_M_getloc () const  [inline]
```

Locale access.

Returns

A reference to the current locale.

Like getloc above, but returns a reference instead of generating a copy.
Referenced by std::money_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter
std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIte
std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::num_put< _CharT, _OutIter
std::time_put< _CharT, _OutIter >::do_put(), std::time_get< _CharT, _InIter >::get(), and std::time_put< _CharT, _OutIter >::put().

flags() [1/2]

```
fmtflags std::ios_base::flags () const  [inline], [nodiscard]
```

Access to format flags.

Returns

The format control flags for both input and output.

Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >
std::num_put< _CharT, _OutIter >::do_put(), std::num_put< _CharT, _OutIter >::do_put(), std::chrono::operator<<(),
std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator<<,
std::__detail::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(), std::operator>>(),
std::operator>>(), std::operator>>(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB

flags() [2/2]

```
fmtflags std::ios_base::flags (
            fmtflags __fmtfl)  [inline]
```

Setting new format flags all at once.

Parameters

__fmtfl	The new flags to set.

Returns

The previous format control flags.

This function overwrites all the format flags with *__fmtfl*.

getloc()

```
locale std::ios_base::getloc () const  [inline], [nodiscard]
```

Locale access.

Returns

A copy of the current locale.

If `imbue(loc)` has previously been called, then this function returns `loc`. Otherwise, it returns a copy of `std::locale()`, the global C++ locale.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::money_put< _CharT, _OutIter >::do_put(), std::basic_ios< _CharT, _Tr
std::chrono::operator<<(), std::operator>>(), std::operator>>(), and std::ws().

imbue()

```
locale std::ios_base::imbue (
            const locale & __loc) throw ( )
```

Setting a new locale.

Parameters

__loc	The new locale.

Returns

The previous locale.

Sets the new locale for this stream, and then invokes each callback with imbue_event.
Referenced by std::basic_ios< _CharT, _Traits >::imbue().

iword()

```
long & std::ios_base::iword (
            int __ix)  [inline]
```

Access to integer array.

Parameters

__ix	Index into the array.

Returns

A reference to an integer associated with the index.

The iword function provides access to an array of integers that can be used for any purpose. The array grows as required to hold the supplied index. All integers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.
References iword().
Referenced by iword().

precision() [1/2]

```
streamsize std::ios_base::precision () const  [inline], [nodiscard]
```
Flags access.

Returns

The precision to generate on certain output operations.

Be careful if you try to give a definition of *precision* here; see DR 189.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), and std::chrono::operator<<().

precision() [2/2]

```
streamsize std::ios_base::precision (
            streamsize __prec)  [inline]
```
Changing flags.

Parameters

__prec	The new precision value.

Returns

The previous value of precision().

pword()

```
void *& std::ios_base::pword (
            int __ix)  [inline]
```
Access to void pointer array.

Parameters

_↩ _ix	Index into the array.

Returns

A reference to a void* associated with the index.

The pword function provides access to an array of pointers that can be used for any purpose. The array grows as required to hold the supplied index. All pointers in the array are initialized to 0.
The implementation reserves several indices. You should use xalloc to obtain an index that is safe to use. Also note that since the array can grow dynamically, it is not safe to hold onto the reference.

register_callback()

```
void std::ios_base::register_callback (
            event_callback __fn,
            int __index)
```
Add the callback __fn with parameter __index.

Parameters

__fn	The function to add.
__index	The integer to pass to the function when invoked.

Registers a function as an event callback with an integer parameter to be passed to the function when invoked. Multiple copies of the function are allowed. If there are multiple callbacks, they are invoked in the order they were registered.

setf() [1/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl)  [inline]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.

Returns

The previous format control flags.

This function sets additional flags in format control. Flags that were previously set remain set.
Referenced by std::__detail::operator>>().

setf() [2/2]

```
fmtflags std::ios_base::setf (
            fmtflags __fmtfl,
            fmtflags __mask)  [inline]
```

Setting new format flags.

Parameters

__fmtfl	Additional flags to set.
__mask	The flags mask for *fmtfl*.

Returns

The previous format control flags.

This function clears *mask* in the format flags, then sets *fmtfl* `&` *mask*. An example mask is `ios_base::adjustfield`.

sync_with_stdio()

```
bool std::ios_base::sync_with_stdio (
            bool __sync = true)  [static]
```

Interaction with the standard C I/O objects.

Parameters

__sync	Whether to synchronize or not.

Returns

True if the standard streams were previously synchronized.

The synchronization referred to is *only* that between the standard C facilities (e.g., stdout) and the standard C++ objects (e.g., cout). User-declared streams are unaffected. See https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary

unsetf()

```
void std::ios_base::unsetf (
            fmtflags __mask)  [inline]
```

Clearing format flags.

Parameters

__mask	The flags to unset.

This function clears *__mask* in the format flags.

width() [1/2]

```
streamsize std::ios_base::width () const  [inline], [nodiscard]
```

Flags access.

Returns

The minimum field width to generate on output operations.

Minimum field width refers to the number of characters.
Referenced by std::basic_ios< _CharT, _Traits >::copyfmt(), std::num_put< _CharT, _OutIter >::do_put(), std::operator>>(), and std::operator>>().

width() [2/2]

```
streamsize std::ios_base::width (
            streamsize __wide)  [inline]
```

Changing flags.

Parameters

__wide	The new width value.

Returns

The previous value of width().

xalloc()

```
int std::ios_base::xalloc () throw ( )   [static]
```
Access to unique indices.

Returns

An integer different from all previous calls.

This function returns a unique integer every time it is called. It can be used for any purpose, but is primarily intended to be a unique index for the iword and pword functions. The expectation is that an application calls xalloc in order to obtain an index in the iword and pword arrays that can be used without fear of conflict.
The implementation maintains a static variable that is incremented and returned on each invocation. xalloc is guaranteed to return an index that is safe to use in the iword and pword arrays.

6.554.6 Member Data Documentation

adjustfield

```
const fmtflags std::ios_base::adjustfield  [static]
```
A mask of left|right|internal. Useful for the 2-arg form of `setf`.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::internal(), std::left(), and std::right().

app

```
const openmode std::ios_base::app  [static]
```
Seek to end before each write.
Referenced by std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_filebuf< _CharT, _Traits >::overflow(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

ate

```
const openmode std::ios_base::ate  [static]
```
Open and seek to end immediately after opening.
Referenced by std::basic_filebuf< _CharT, _Traits >::open().

badbit

```
const iostate std::ios_base::badbit  [static]
```
Indicates a loss of integrity in an input or output sequence (such as an irrecoverable read error from a file).
Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::basic_ios< _CharT, _Traits >::clear(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_typ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >:: std::basic_istream< _CharT, _Traits >::tellg(), std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

basefield

```
const fmtflags std::ios_base::basefield  [static]
```
A mask of dec|oct|hex. Useful for the 2-arg form of `setf`.
Referenced by std::dec(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), std::hex(), and std::oct().

beg

const seekdir std::ios_base::beg [static]
Request a seek relative to the beginning of the stream.
Referenced by std::basic_filebuf< _CharT, _Traits >::seekpos().

binary

const openmode std::ios_base::binary [static]
Perform input and output in binary mode (as opposed to text mode). This is probably not what you think it is; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/fstreams.html#std.io.filestreams.binary.
Referenced by std::basic_filebuf< _CharT, _Traits >::showmanyc().

boolalpha

const fmtflags std::ios_base::boolalpha [static]
Insert/extract bool in alphabetic rather than numeric format.
Referenced by std::boolalpha(), std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::noboolalpha().

cur

const seekdir std::ios_base::cur [static]
Request a seek relative to the current position within the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, std::basic_istream< _CharT, _Traits >::tellg(), and std::basic_ostream< _CharT, _Traits >::tellp().

dec

const fmtflags std::ios_base::dec [static]
Converts integer input or generates integer output in decimal base.
Referenced by std::dec(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::opera and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>.

end

const seekdir std::ios_base::end [static]
Request a seek relative to the current end of the sequence.
Referenced by std::basic_filebuf< _CharT, _Traits >::open(), and std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff().

eofbit

const iostate std::ios_base::eofbit [static]
Indicates that an input operation reached the end of an input sequence.
Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >::do_get(), std::num_get< _CharT, _InIter >:: std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_date(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::do_get_time(), std::time_get< _CharT, _InIter std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Tra std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >::readsome(), std::basic_iostream< char_type, t std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, char_traits std::basic_istream< _CharT, _Traits >::unget(), and std::ws().

failbit

```
const iostate std::ios_base::failbit [static]
```

Indicates that an input operation failed to read the expected characters, or that an output operation failed to generate the desired characters.

Referenced by std::basic_ostream< _CharT, _Traits >::sentry::sentry(), std::basic_fstream< char >::close(), std::basic_ifstream< char > std::basic_ofstream< char >::close(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname std::time_get< _CharT, _InIter >::do_get_weekday(), std::time_get< _CharT, _InIter >::do_get_year(), std::basic_iostream< char_type, std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(), std::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_ostream< _CharT, _Traits > and std::basic_ostream< _CharT, _Traits >::seekp().

fixed

```
const fmtflags std::ios_base::fixed [static]
```

Generate floating-point output in fixed-point notation.

Referenced by std::fixed(), and std::hexfloat().

floatfield

```
const fmtflags std::ios_base::floatfield [static]
```

A mask of scientific|fixed. Useful for the 2-arg form of `setf`.

Referenced by std::defaultfloat(), std::fixed(), std::hexfloat(), and std::scientific().

goodbit

```
const iostate std::ios_base::goodbit [static]
```

Indicates all is well.

Referenced by std::basic_istream< _CharT, _Traits >::sentry::sentry(), std::num_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get(), std::time_get< _CharT, _InIter >::do_get_monthname(), std::time_get< _CharT, _InIter >::d std::time_get< _CharT, _InIter >::do_get_year(), std::basic_ostream< _CharT, _Traits >::flush(), std::basic_istream< _CharT, _Traits >: std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::time_get< _CharT, _InIter >::get(), std::basic_istream< _CharT, _Traits >::getline(), std::basic_istream< _CharT, _Traits >::ignore() std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_iostream< char_type, traits_ std::basic_ostream< _CharT, _Traits >::operator<<(), std::basic_iostream< char_type, traits_type >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::basic_istream< _CharT, _Traits >::operator>>(), std::operator>>(), std::basic_ostream< _CharT, _Traits >::put(), std::basic_istream< _CharT, _Traits >::putback(), std::basic_istream< _CharT, _Traits >:: std::basic_iostream< char_type, traits_type >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _T std::basic_istream< _CharT, _Traits >::sync(), std::basic_istream< _CharT, char_traits< _CharT > >::tellg(), std::basic_istream< _Cha and std::ws().

hex

```
const fmtflags std::ios_base::hex [static]
```

Converts integer input or generates integer output in hexadecimal base.

Referenced by std::num_get< _CharT, _InIter >::do_get(), std::num_put< _CharT, _OutIter >::do_put(), and std::hex().

in

```
const openmode std::ios_base::in [static]
```

Open for input. Default for `ifstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstream(), std::basic_ifstream< char >::basic_ifstrea
std::basic_istringstream< char >::basic_istringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char >
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ifstream< char >::open(), std::basic_ifstream< char >::open(),
std::basic_ifstream< char >::open(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::p
std::basic_streambuf< char_type, traits_type >::pubseekoff(), std::basic_streambuf< char_type, traits_type >::pubseekpos(),
std::basic_istream< _CharT, _Traits >::seekg(), std::basic_istream< _CharT, _Traits >::seekg(), std::basic_streambuf< char_type, traits
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_streambuf< char_type, trait
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_stringbuf< char >::seekpos(), std::basic_filebuf< _CharT, _Traits >
std::basic_stringbuf< char >::showmanyc(), std::basic_istream< _CharT, _Traits >::tellg(), std::basic_filebuf< _CharT, _Traits >::underfl
std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

internal

```
const fmtflags std::ios_base::internal  [static]
```

Adds fill characters at a designated internal point in certain generated output, or identical to `right` if no such point is designated.
Referenced by std::internal().

left

```
const fmtflags std::ios_base::left  [static]
```

Adds fill characters on the right (final positions) of certain generated output. (I.e., the thing you print is flush left.).
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::left(), and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x

oct

```
const fmtflags std::ios_base::oct  [static]
```

Converts integer input or generates integer output in octal base.
Referenced by std::oct().

out

```
const openmode std::ios_base::out  [static]
```

Open for output. Default for `ofstream` and fstream.
Referenced by std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::basic_fstream(), std::basic_fstream< char >::b
std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofstream(), std::basic_ofstream< char >::basic_ofs
std::basic_ostringstream< char >::basic_ostringstream(), std::basic_stringbuf< char >::basic_stringbuf(), std::basic_stringstream< char
std::basic_filebuf< char_type, traits_type >::_M_set_buffer(), std::basic_fstream< char >::open(), std::basic_fstream< char >::open(),
std::basic_fstream< char >::open(), std::basic_ofstream< char >::open(), std::basic_ofstream< char >::open(),
std::basic_ofstream< char >::open(), std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::o
std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::pubseekoff(),
std::basic_streambuf< char_type, traits_type >::pubseekpos(), std::basic_streambuf< char_type, traits_type >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< char >::seekoff(), std::basic_ostream< _CharT, _Traits >
std::basic_ostream< _CharT, _Traits >::seekp(), std::basic_streambuf< char_type, traits_type >::seekpos(), std::basic_stringbuf< _Cha
std::basic_stringbuf< char >::seekpos(), std::basic_ostream< _CharT, _Traits >::tellp(), and std::basic_filebuf< _CharT, _Traits >::xsput

right

```
const fmtflags std::ios_base::right  [static]
```

Adds fill characters on the left (initial positions) of certain generated output. (I.e., the thing you print is flush right.).
Referenced by std::right().

scientific

```
const fmtflags std::ios_base::scientific  [static]
```
Generates floating-point output in scientific notation.
Referenced by std::hexfloat(), and std::scientific().

showbase

```
const fmtflags std::ios_base::showbase  [static]
```
Generates a prefix indicating the numeric base of generated integer output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::noshowbase(), and std::showbase().

showpoint

```
const fmtflags std::ios_base::showpoint  [static]
```
Generates a decimal-point character unconditionally in generated floating-point output.
Referenced by std::noshowpoint(), and std::showpoint().

showpos

```
const fmtflags std::ios_base::showpos  [static]
```
Generates a + sign in non-negative generated numeric output.
Referenced by std::noshowpos(), and std::showpos().

skipws

```
const fmtflags std::ios_base::skipws  [static]
```
Skips leading white space before certain input operations.
Referenced by std::noskipws(), std::__detail::operator>>(), std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0x
and std::skipws().

trunc

```
const openmode std::ios_base::trunc  [static]
```
Truncate an existing stream when opening. Default for `ofstream`.

unitbuf

```
const fmtflags std::ios_base::unitbuf  [static]
```
Flushes output after each output operation.
Referenced by std::basic_ostream< _CharT, _Traits >::sentry::~sentry(), std::nounitbuf(), and std::unitbuf().

uppercase

```
const fmtflags std::ios_base::uppercase  [static]
```
Replaces certain lowercase letters with their uppercase equivalents in generated output.
Referenced by std::num_put< _CharT, _OutIter >::do_put(), std::nouppercase(), and std::uppercase().
The documentation for this class was generated from the following file:

- ios_base.h

6.555 std::is_abstract< _Tp > Struct Template Reference

```
#include <type_traits>
```

6.555.1 Detailed Description

template<**typename _Tp**>
struct std::is_abstract< **_Tp** >

is_abstract
The documentation for this struct was generated from the following file:

- type_traits

6.556 std::is_arithmetic< _Tp > Struct Template Reference

`#include <type_traits>`

6.556.1 Detailed Description

template<**typename _Tp**>
struct std::is_arithmetic< **_Tp** >

is_arithmetic
The documentation for this struct was generated from the following file:

- type_traits

6.557 std::is_array< _Tp > Struct Template Reference

`#include <type_traits>`

6.557.1 Detailed Description

template<**typename _Tp**>
struct std::is_array< **_Tp** >

is_array
The documentation for this struct was generated from the following file:

- type_traits

6.558 std::is_assignable< _Tp, _Up > Struct Template Reference

`#include <type_traits>`

6.558.1 Detailed Description

template<**typename _Tp, typename _Up**>
struct std::is_assignable< **_Tp, _Up** >

is_assignable
The documentation for this struct was generated from the following file:

- type_traits

6.559 std::is_base_of< _Base, _Derived > Struct Template Reference

`#include <type_traits>`

6.559.1 Detailed Description

template<typename _Base, typename _Derived>
struct std::is_base_of< _Base, _Derived >

is_base_of
The documentation for this struct was generated from the following file:

- type_traits

6.560 std::is_bind_expression< _Tp > Struct Template Reference

`#include <functional>`

6.560.1 Detailed Description

template<typename _Tp>
struct std::is_bind_expression< _Tp >

Trait that identifies a bind expression.
Determines if the given type `_Tp` is a function object that should be treated as a subexpression when evaluating calls to function objects returned by `std::bind`.
C++11 [func.bind.isbind].

Since

C++11

The documentation for this struct was generated from the following file:

- functional

6.561 std::is_bind_expression< _Bind< _Signature > > Struct Template Reference

`#include <functional>`

6.561.1 Detailed Description

template<typename _Signature>
struct std::is_bind_expression< _Bind< _Signature > >

Class template _Bind is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

6.562 std::is_bind_expression< _Bind_result< _Result, _Signature > > Struct Template Reference

`#include <functional>`

6.562.1 Detailed Description

template<typename _Result, typename _Signature>
struct std::is_bind_expression< _Bind_result< _Result, _Signature > >

Class template _Bind_result is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

6.563 std::is_bind_expression< const _Bind< _Signature > > Struct Template Reference

`#include <functional>`

6.563.1 Detailed Description

template<typename _Signature>
struct std::is_bind_expression< const _Bind< _Signature > >

Class template _Bind is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

6.564 std::is_bind_expression< const _Bind_result< _Result, _Signature > > Struct Template Reference

`#include <functional>`

6.564.1 Detailed Description

template<typename _Result, typename _Signature>
struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >

Class template _Bind_result is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

6.565 std::is_bind_expression< const volatile _Bind< _Signature > > Struct Template Reference

`#include <functional>`

6.565.1 Detailed Description

template<typename _Signature>
struct std::is_bind_expression< const volatile _Bind< _Signature > >

Class template _Bind is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

6.566 std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > > Struct Template Reference

`#include <functional>`

6.566.1 Detailed Description

template<typename _Result, typename _Signature>
struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >

Class template _Bind_result is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

6.567 std::is_bind_expression< volatile _Bind< _Signature > > Struct Template Reference

`#include <functional>`

6.567.1 Detailed Description

template<typename _Signature>
struct std::is_bind_expression< volatile _Bind< _Signature > >

Class template _Bind is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

6.568 std::is_bind_expression< volatile _Bind_result< _Result, _Signature > > Struct Template Reference

`#include <functional>`

6.568.1 Detailed Description

template<typename _Result, typename _Signature>
struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >

Class template _Bind_result is always a bind expression.
The documentation for this struct was generated from the following file:

- functional

6.569 std::is_class< _Tp > Struct Template Reference

`#include <type_traits>`

6.569.1 Detailed Description

template<typename _Tp>
struct std::is_class< _Tp >

is_class
The documentation for this struct was generated from the following file:

- type_traits

6.570 std::is_compound< _Tp > Struct Template Reference

`#include <type_traits>`

6.570.1 Detailed Description

template<typename _Tp>
struct std::is_compound< _Tp >

is_compound
The documentation for this struct was generated from the following file:

- type_traits

6.571 std::is_const< _Tp > Struct Template Reference

`#include <type_traits>`

6.571.1 Detailed Description

template<typename _Tp>
struct std::is_const< _Tp >

is_const
The documentation for this struct was generated from the following file:

- type_traits

6.572 std::is_constructible< _Tp, _Args > Struct Template Reference

`#include <type_traits>`

6.572.1 Detailed Description

template<typename _Tp, typename... _Args>
struct std::is_constructible< _Tp, _Args >

is_constructible
The documentation for this struct was generated from the following file:

- type_traits

6.573 std::is_copy_assignable< _Tp > Struct Template Reference

`#include <type_traits>`

6.573.1 Detailed Description

template<typename _Tp>
struct std::is_copy_assignable< _Tp >

is_copy_assignable
The documentation for this struct was generated from the following file:

- type_traits

6.574 std::is_copy_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

6.574.1 Detailed Description

template<typename _Tp>
struct std::is_copy_constructible< _Tp >

is_copy_constructible
The documentation for this struct was generated from the following file:

- type_traits

6.575 std::is_default_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

6.575.1 Detailed Description

template<typename _Tp>
struct std::is_default_constructible< _Tp >

is_default_constructible
The documentation for this struct was generated from the following file:

- type_traits

6.576 std::is_destructible< _Tp > Struct Template Reference

`#include <type_traits>`

6.576.1 Detailed Description

template<typename _Tp>
struct std::is_destructible< _Tp >

is_destructible
The documentation for this struct was generated from the following file:

- type_traits

6.577 std::is_empty< _Tp > Struct Template Reference

`#include <type_traits>`

6.577.1 Detailed Description

template<typename _Tp>
struct std::is_empty< _Tp >

is_empty
The documentation for this struct was generated from the following file:

- type_traits

6.578 std::is_enum< _Tp > Struct Template Reference

`#include <type_traits>`

6.578.1 Detailed Description

template<typename _Tp>
struct std::is_enum< _Tp >

is_enum
The documentation for this struct was generated from the following file:

- type_traits

6.579 std::is_error_code_enum< _Tp > Struct Template Reference

`#include <system_error>`

6.579.1 Detailed Description

template<typename _Tp>
struct std::is_error_code_enum< _Tp >

is_error_code_enum
The documentation for this struct was generated from the following file:

- system_error

6.580 std::is_error_code_enum< future_errc > Struct Reference

`#include <future>`

6.580.1 Detailed Description

Specialization that allows `future_errc` to convert to `error_code`.
The documentation for this struct was generated from the following file:

- future

6.581 std::is_error_condition_enum< _Tp > Struct Template Reference

`#include <system_error>`

6.581.1 Detailed Description

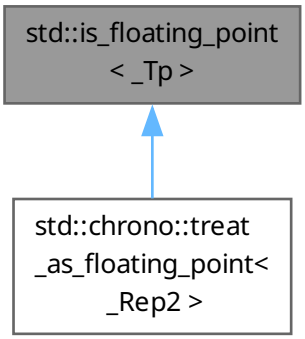
template<typename _Tp>
struct std::is_error_condition_enum< _Tp >

is_error_condition_enum
The documentation for this struct was generated from the following file:

- system_error

6.582 std::is_floating_point< _Tp > Struct Template Reference

`#include <type_traits>`
Inheritance diagram for std::is_floating_point< _Tp >:



6.582.1 Detailed Description

template<typename _Tp>
struct std::is_floating_point< _Tp >

is_floating_point
The documentation for this struct was generated from the following file:

- type_traits

6.583 std::is_function< _Tp > Struct Template Reference

`#include <type_traits>`

6.583.1 Detailed Description

template<typename _Tp>
struct std::is_function< _Tp >

is_function
The documentation for this struct was generated from the following file:

- type_traits

6.584 std::is_fundamental< _Tp > Struct Template Reference

`#include <type_traits>`

6.584.1 Detailed Description

template<typename _Tp>
struct std::is_fundamental< _Tp >

is_fundamental
The documentation for this struct was generated from the following file:

- type_traits

6.585 std::is_integral< _Tp > Struct Template Reference

`#include <type_traits>`

6.585.1 Detailed Description

template<typename _Tp>
struct std::is_integral< _Tp >

is_integral
The documentation for this struct was generated from the following file:

- type_traits

6.586 std::is_layout_compatible< _Tp, _Up > Struct Template Reference

`#include <type_traits>`

6.586.1 Detailed Description

template<typename _Tp, typename _Up>
struct std::is_layout_compatible< _Tp, _Up >

- Remove references and cv-qualifiers.

  Since

  C++20
  C++20

The documentation for this struct was generated from the following file:

- type_traits

6.587 std::is_literal_type< _Tp > Struct Template Reference

`#include <type_traits>`

6.587.1 Detailed Description

template<typename _Tp>
struct std::is_literal_type< _Tp >

is_literal_type

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

The documentation for this struct was generated from the following file:

- type_traits

6.588 std::is_lvalue_reference< typename > Struct Template Reference

`#include <type_traits>`

6.588.1 Detailed Description

template<typename>
struct std::is_lvalue_reference< typename >

is_lvalue_reference
The documentation for this struct was generated from the following file:

- type_traits

6.589 std::is_member_function_pointer< _Tp > Struct Template Reference

`#include <type_traits>`

6.589.1 Detailed Description

template<typename _Tp>
struct std::is_member_function_pointer< _Tp >

is_member_function_pointer
The documentation for this struct was generated from the following file:

- type_traits

6.590 std::is_member_object_pointer< _Tp > Struct Template Reference

`#include <type_traits>`

6.590.1 Detailed Description

template<typename _Tp>
struct std::is_member_object_pointer< _Tp >

is_member_object_pointer
The documentation for this struct was generated from the following file:

- type_traits

6.591 std::is_member_pointer< _Tp > Struct Template Reference

`#include <type_traits>`

6.591.1 Detailed Description

template<typename _Tp>
struct std::is_member_pointer< _Tp >

is_member_pointer
The documentation for this struct was generated from the following file:

- type_traits

6.592 std::is_move_assignable< _Tp > Struct Template Reference

`#include <type_traits>`

6.592.1 Detailed Description

template<typename _Tp>
struct std::is_move_assignable< _Tp >

is_move_assignable
The documentation for this struct was generated from the following file:

- type_traits

6.593 std::is_move_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

6.593.1 Detailed Description

template<typename _Tp>
struct std::is_move_constructible< _Tp >

is_move_constructible
The documentation for this struct was generated from the following file:

- type_traits

6.594 std::is_nothrow_assignable< _Tp, _Up > Struct Template Reference

`#include <type_traits>`

6.594.1 Detailed Description

template<typename _Tp, typename _Up>
struct std::is_nothrow_assignable< _Tp, _Up >

is_nothrow_assignable
The documentation for this struct was generated from the following file:

- type_traits

6.595 std::is_nothrow_constructible< _Tp, _Args > Struct Template Reference

`#include <type_traits>`

6.595.1 Detailed Description

template<typename _Tp, typename... _Args>
struct std::is_nothrow_constructible< _Tp, _Args >

is_nothrow_constructible
The documentation for this struct was generated from the following file:

- type_traits

6.596 std::is_nothrow_copy_assignable< _Tp > Struct Template Reference

`#include <type_traits>`

6.596.1 Detailed Description

template<typename _Tp>
struct std::is_nothrow_copy_assignable< _Tp >

is_nothrow_copy_assignable
The documentation for this struct was generated from the following file:

- type_traits

6.597 std::is_nothrow_copy_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

6.597.1 Detailed Description

template<typename _Tp>
struct std::is_nothrow_copy_constructible< _Tp >

is_nothrow_copy_constructible
The documentation for this struct was generated from the following file:

- type_traits

6.598 std::is_nothrow_default_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

6.598.1 Detailed Description

template<typename _Tp>
struct std::is_nothrow_default_constructible< _Tp >

is_nothrow_default_constructible
The documentation for this struct was generated from the following file:

- type_traits

6.599 std::is_nothrow_destructible< _Tp > Struct Template Reference

`#include <type_traits>`

6.599.1 Detailed Description

template<typename _Tp>
struct std::is_nothrow_destructible< _Tp >

is_nothrow_destructible
The documentation for this struct was generated from the following file:

- type_traits

6.600 std::is_nothrow_move_assignable< _Tp > Struct Template Reference

`#include <type_traits>`

6.600.1 Detailed Description

template<typename _Tp>
struct std::is_nothrow_move_assignable< _Tp >

is_nothrow_move_assignable
The documentation for this struct was generated from the following file:

- type_traits

6.601 std::is_nothrow_move_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

6.601.1 Detailed Description

template<typename _Tp>
struct std::is_nothrow_move_constructible< _Tp >

is_nothrow_move_constructible
The documentation for this struct was generated from the following file:

- type_traits

6.602 std::is_object< _Tp > Struct Template Reference

`#include <type_traits>`

6.602.1 Detailed Description

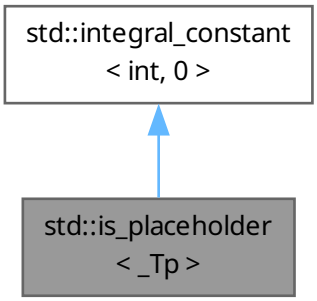
template<**typename _Tp**>
struct std::is_object< **_Tp** >

is_object
The documentation for this struct was generated from the following file:

- type_traits

6.603 std::is_placeholder< _Tp > Struct Template Reference

`#include <functional>`
Inheritance diagram for std::is_placeholder< _Tp >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr int **value**

6.603.1 Detailed Description

template<**typename _Tp**>
struct std::is_placeholder< **_Tp** >

Determines if the given type _Tp is a placeholder in a bind() expression and, if so, which placeholder it is.
C++11 [func.bind.isplace].

Since

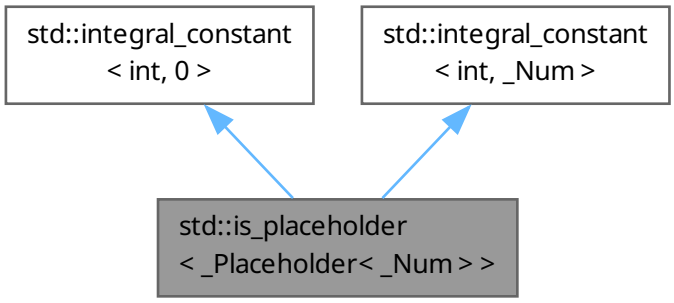
C++11

The documentation for this struct was generated from the following file:

- functional

6.604 std::is_placeholder< _Placeholder< _Num > > Struct Template Reference

`#include <functional>`
Inheritance diagram for std::is_placeholder< _Placeholder< _Num > >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr int **value**

6.604.1 Detailed Description

template<**int _Num**>
struct std::is_placeholder< **_Placeholder**< **_Num** > >

Partial specialization of is_placeholder that provides the placeholder number for the placeholder objects defined by libstdc++.

Since

C++11

The documentation for this struct was generated from the following file:

- functional

6.605 std::is_pod< _Tp > Struct Template Reference

`#include <type_traits>`

6.605.1 Detailed Description

template<typename _Tp>
struct std::is_pod< _Tp >

is_pod

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

The documentation for this struct was generated from the following file:

- type_traits

6.606 std::is_pointer< _Tp > Struct Template Reference

`#include <type_traits>`

6.606.1 Detailed Description

template<typename _Tp>
struct std::is_pointer< _Tp >

is_pointer
The documentation for this struct was generated from the following file:

- type_traits

6.607 std::is_pointer_interconvertible_base_of< _Base, _Derived > Struct Template Reference

`#include <type_traits>`

6.607.1 Detailed Description

template<typename _Base, typename _Derived>
struct std::is_pointer_interconvertible_base_of< _Base, _Derived >

True if `_Derived` is standard-layout and has a base class of type `_Base`.

Since

C++20

The documentation for this struct was generated from the following file:

- type_traits

6.608 std::is_polymorphic< _Tp > Struct Template Reference

`#include <type_traits>`

6.608.1 Detailed Description

template<typename _Tp>
struct std::is_polymorphic< _Tp >

is_polymorphic
The documentation for this struct was generated from the following file:

- type_traits

6.609 std::is_reference< _Tp > Struct Template Reference

`#include <type_traits>`

6.609.1 Detailed Description

template<typename _Tp>
struct std::is_reference< _Tp >

is_reference
The documentation for this struct was generated from the following file:

- type_traits

6.610 std::is_rvalue_reference< typename > Struct Template Reference

`#include <type_traits>`

6.610.1 Detailed Description

template<typename>
struct std::is_rvalue_reference< typename >

is_rvalue_reference
The documentation for this struct was generated from the following file:

- type_traits

6.611 std::is_same< _Tp, _Up > Struct Template Reference

`#include <type_traits>`

6.611.1 Detailed Description

template<typename _Tp, typename _Up>
struct std::is_same< _Tp, _Up >

is_same
The documentation for this struct was generated from the following file:

- type_traits

6.612 std::is_scalar< _Tp > Struct Template Reference

`#include <type_traits>`

6.612.1 Detailed Description

template<typename _Tp>
struct std::is_scalar< _Tp >

is_scalar
The documentation for this struct was generated from the following file:

- type_traits

6.613 std::is_signed< _Tp > Struct Template Reference

`#include <type_traits>`

6.613.1 Detailed Description

template<typename _Tp>
struct std::is_signed< _Tp >

is_signed
The documentation for this struct was generated from the following file:

- type_traits

6.614 std::is_standard_layout< _Tp > Struct Template Reference

`#include <type_traits>`

6.614.1 Detailed Description

template<typename _Tp>
struct std::is_standard_layout< _Tp >

is_standard_layout
The documentation for this struct was generated from the following file:

- type_traits

6.615 std::is_trivial< _Tp > Struct Template Reference

`#include <type_traits>`

6.615.1 Detailed Description

template<typename _Tp>
struct std::is_trivial< _Tp >

is_trivial

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

The documentation for this struct was generated from the following file:

- type_traits

6.616 std::is_trivially_assignable< _Tp, _Up > Struct Template Reference

`#include <type_traits>`

6.616.1 Detailed Description

template<typename _Tp, typename _Up>
struct std::is_trivially_assignable< _Tp, _Up >

is_trivially_assignable
The documentation for this struct was generated from the following file:

- type_traits

6.617 std::is_trivially_constructible< _Tp, _Args > Struct Template Reference

`#include <type_traits>`

6.617.1 Detailed Description

template<typename _Tp, typename... _Args>
struct std::is_trivially_constructible< _Tp, _Args >

is_trivially_constructible
The documentation for this struct was generated from the following file:

- type_traits

6.618 std::is_trivially_copy_assignable< _Tp > Struct Template Reference

`#include <type_traits>`

6.618.1 Detailed Description

template<typename _Tp>
struct std::is_trivially_copy_assignable< _Tp >

is_trivially_copy_assignable
The documentation for this struct was generated from the following file:

- type_traits

6.619 std::is_trivially_copy_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

6.619.1 Detailed Description

template<typename _Tp>
struct std::is_trivially_copy_constructible< _Tp >

is_trivially_copy_constructible
The documentation for this struct was generated from the following file:

- type_traits

6.620 std::is_trivially_copyable< _Tp > Struct Template Reference

`#include <type_traits>`

6.620.1 Detailed Description

template<**typename _Tp**>
struct std::is_trivially_copyable< **_Tp** >

is_trivially_copyable
The documentation for this struct was generated from the following file:

- type_traits

6.621 std::is_trivially_default_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

6.621.1 Detailed Description

template<**typename _Tp**>
struct std::is_trivially_default_constructible< **_Tp** >

is_trivially_default_constructible
The documentation for this struct was generated from the following file:

- type_traits

6.622 std::is_trivially_destructible< _Tp > Struct Template Reference

`#include <type_traits>`

6.622.1 Detailed Description

template<**typename _Tp**>
struct std::is_trivially_destructible< **_Tp** >

is_trivially_destructible
The documentation for this struct was generated from the following file:

- type_traits

6.623 std::is_trivially_move_assignable< _Tp > Struct Template Reference

`#include <type_traits>`

6.623.1 Detailed Description

template<**typename _Tp**>
struct std::is_trivially_move_assignable< **_Tp** >

is_trivially_move_assignable
The documentation for this struct was generated from the following file:

- type_traits

6.624 std::is_trivially_move_constructible< _Tp > Struct Template Reference

`#include <type_traits>`

6.624.1 Detailed Description

template<typename _Tp>
struct std::is_trivially_move_constructible< _Tp >

is_trivially_move_constructible
The documentation for this struct was generated from the following file:

- type_traits

6.625 std::is_union< _Tp > Struct Template Reference

`#include <type_traits>`

6.625.1 Detailed Description

template<typename _Tp>
struct std::is_union< _Tp >

is_union
The documentation for this struct was generated from the following file:

- type_traits

6.626 std::is_unsigned< _Tp > Struct Template Reference

`#include <type_traits>`

6.626.1 Detailed Description

template<typename _Tp>
struct std::is_unsigned< _Tp >

is_unsigned
The documentation for this struct was generated from the following file:

- type_traits

6.627 std::is_void< _Tp > Struct Template Reference

`#include <type_traits>`

6.627.1 Detailed Description

template<typename _Tp>
struct std::is_void< _Tp >

is_void
The documentation for this struct was generated from the following file:

- type_traits

6.628 std::is_volatile< _Tp > Struct Template Reference

`#include <type_traits>`

6.628.1 Detailed Description

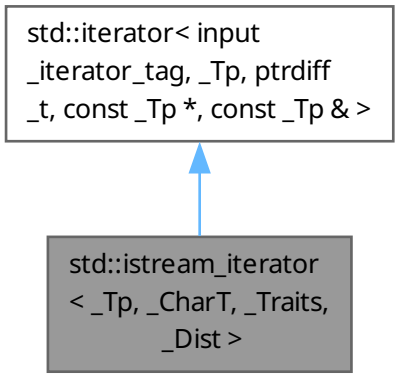
template<**typename _Tp**>
struct std::is_volatile< **_Tp** >

is_volatile
The documentation for this struct was generated from the following file:

- type_traits

6.629 std::istream_iterator< _Tp, _CharT, _Traits, _Dist > Class Template Reference

`#include <stream_iterator.h>`
Inheritance diagram for std::istream_iterator< _Tp, _CharT, _Traits, _Dist >:



Public Types

- typedef _CharT **char_type**
- typedef ptrdiff_t difference_type
- typedef basic_istream< _CharT, _Traits > **istream_type**
- typedef input_iterator_tag iterator_category
- typedef const _Tp * pointer
- typedef const _Tp & reference
- typedef _Traits **traits_type**
- typedef _Tp value_type

Public Member Functions

- constexpr istream_iterator () noexcept(/*conditional */)
- constexpr **istream_iterator** (const istream_iterator &__obj) noexcept(/*conditional */)
- constexpr **istream_iterator** (default_sentinel_t) noexcept(is_nothrow_default_constructible_v< _Tp >)
- istream_iterator (istream_type &__s)
- const _Tp & **operator*** () const noexcept
- istream_iterator & **operator++** ()
- istream_iterator **operator++** (int)

- const _Tp * **operator->** () const noexcept
- istream_iterator & **operator=** (const istream_iterator &)=default

Friends

- bool **operator==** (const istream_iterator &__i, default_sentinel_t) noexcept
- bool operator== (const istream_iterator &__x, const istream_iterator &__y) noexcept

6.629.1 Detailed Description

template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename _Dist = ptrdiff_t>
class std::istream_iterator< _Tp, _CharT, _Traits, _Dist >

Provides input iterator semantics for streams.

6.629.2 Member Typedef Documentation

difference_type

```
typedef ptrdiff_t std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp &
>::difference_type  [inherited]
```

Distance between iterators is represented as this type.

iterator_category

```
typedef input_iterator_tag std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const
_Tp & >::iterator_category  [inherited]
```

One of the tag types.

pointer

```
typedef const _Tp * std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp &
>::pointer  [inherited]
```

This type represents a pointer-to-value_type.

reference

```
typedef const _Tp & std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp &
>::reference  [inherited]
```

This type represents a reference-to-value_type.

value_type

```
typedef _Tp std::iterator< input_iterator_tag, _Tp, ptrdiff_t, const _Tp *, const _Tp & >::↩
value_type  [inherited]
```

The type "pointed to" by the iterator.

6.629.3 Constructor & Destructor Documentation

istream_iterator() [1/2]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator ()  [inline], [constexpr],
[noexcept]
```

Construct end of input stream iterator.
Referenced by operator==.

istream_iterator() [2/2]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >::istream_iterator (
            istream_type & __s)  [inline]
```

Construct start of input stream iterator.
References std::__addressof().

6.629.4 Friends And Related Symbol Documentation

operator==

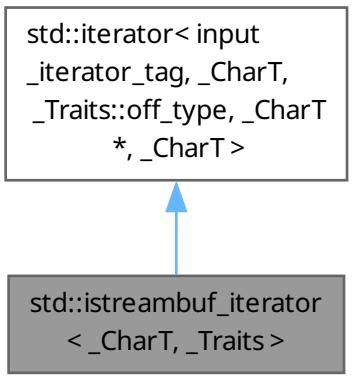
```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>, typename
_Dist = ptrdiff_t>
bool operator== (
            const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __x,
            const istream_iterator< _Tp, _CharT, _Traits, _Dist > & __y)  [friend]
```

Return true if the iterators refer to the same stream, or are both at end-of-stream.
References istream_iterator().
The documentation for this class was generated from the following file:

- stream_iterator.h

6.630 std::istreambuf_iterator< _CharT, _Traits > Class Template Reference

`#include <streambuf_iterator.h>`
Inheritance diagram for std::istreambuf_iterator< _CharT, _Traits >:



Public Types

- typedef _Traits::off_type difference_type

- typedef input_iterator_tag iterator_category
- typedef _CharT reference
- typedef _CharT value_type

- using pointer
- typedef _CharT char_type
- typedef _Traits traits_type
- typedef _Traits::int_type int_type
- typedef basic_streambuf< _CharT, _Traits > streambuf_type
- typedef basic_istream< _CharT, _Traits > istream_type

Public Member Functions

- constexpr istreambuf_iterator () noexcept
- **istreambuf_iterator** (const istreambuf_iterator &) noexcept=default
- constexpr **istreambuf_iterator** (default_sentinel_t) noexcept
- istreambuf_iterator (istream_type &__s) noexcept
- istreambuf_iterator (streambuf_type *__s) noexcept
- bool equal (const istreambuf_iterator &__b) const
- char_type operator* () const
- istreambuf_iterator & operator++ ()
- istreambuf_iterator operator++ (int)
- istreambuf_iterator & **operator=** (const istreambuf_iterator &) noexcept=default

Friends

- template<bool _IsMove, typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type **__copy_move_a2** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, _CharT2 *)
- template<typename _CharT2, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, _CharT2 * >::__type **__copy_n_a** (istreambuf_iterator< _CharT2 >, _Size, _CharT2 *, bool)
- template<typename _CharT2, typename _Distance>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, void >::__type **advance** (istreambuf_iterator< _CharT2 > &, _Distance)
- template<typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, ostreambuf_iterator< _CharT2 > >::__type **copy** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, ostreambuf_iterator< _CharT2 >)
- template<typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, istreambuf_iterator< _CharT2 > >::__type **find** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, const _CharT2 &)
- bool **operator==** (const istreambuf_iterator &__i, default_sentinel_t)

6.630.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::istreambuf_iterator< _CharT, _Traits >

Provides input iterator semantics for streambufs.

6.630.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::istreambuf_iterator< _CharT, _Traits >::char_type
```
Public typedefs.

difference_type

```
typedef _Traits::off_type std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *,
_CharT >::difference_type  [inherited]
```
Distance between iterators is represented as this type.

int_type

```
template<typename _CharT, typename _Traits>
typedef _Traits::int_type std::istreambuf_iterator< _CharT, _Traits >::int_type
```
Public typedefs.

istream_type

```
template<typename _CharT, typename _Traits>
typedef basic_istream<_CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::istream_type
```
Public typedefs.

iterator_category

```
typedef input_iterator_tag std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *,
_CharT >::iterator_category  [inherited]
```
One of the tag types.

pointer

```
template<typename _CharT, typename _Traits>
using std::istreambuf_iterator< _CharT, _Traits >::pointer
```
Public typedefs.

reference

```
typedef _CharT std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *, _CharT
>::reference  [inherited]
```
This type represents a reference-to-value_type.

streambuf_type

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<_CharT, _Traits> std::istreambuf_iterator< _CharT, _Traits >::streambuf↩
_type
```
Public typedefs.

traits_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::istreambuf_iterator< _CharT, _Traits >::traits_type
```
Public typedefs.

value_type

```
typedef _CharT std::iterator< input_iterator_tag, _CharT, _Traits::off_type, _CharT *, _CharT
>::value_type  [inherited]
```
The type "pointed to" by the iterator.

6.630.3 Constructor & Destructor Documentation

istreambuf_iterator() [1/3]

```
template<typename _CharT, typename _Traits>
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator ()  [inline], [constexpr], [noexcept]
```
Construct end of input stream iterator.
Referenced by equal(), and operator++().

istreambuf_iterator() [2/3]

```
template<typename _CharT, typename _Traits>
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
            istream_type & __s)  [inline], [noexcept]
```
Construct start of input stream iterator.

istreambuf_iterator() [3/3]

```
template<typename _CharT, typename _Traits>
std::istreambuf_iterator< _CharT, _Traits >::istreambuf_iterator (
            streambuf_type * __s)  [inline], [noexcept]
```
Construct start of streambuf iterator.

6.630.4 Member Function Documentation

equal()

```
template<typename _CharT, typename _Traits>
bool std::istreambuf_iterator< _CharT, _Traits >::equal (
            const istreambuf_iterator< _CharT, _Traits > & __b) const  [inline], [nodiscard]
```
Return true both iterators are end or both are not end.
References istreambuf_iterator().

operator∗()

```
template<typename _CharT, typename _Traits>
char_type std::istreambuf_iterator< _CharT, _Traits >::operator* () const  [inline], [nodiscard]
```
Return the current character pointed to by iterator. This returns streambuf.sgetc(). It cannot be assigned. NB: The result of operator∗() on an end of stream is undefined.

operator++() [1/2]

```
template<typename _CharT, typename _Traits>
istreambuf_iterator & std::istreambuf_iterator< _CharT, _Traits >::operator++ ()  [inline]
```
Advance the iterator. Calls streambuf.sbumpc().

operator++() [2/2]

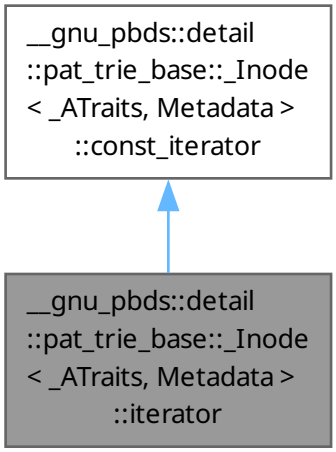
```
template<typename _CharT, typename _Traits>
istreambuf_iterator std::istreambuf_iterator< _CharT, _Traits >::operator++ (
            int )  [inline]
```

Advance the iterator. Calls streambuf.sbumpc().
References istreambuf_iterator().
The documentation for this class was generated from the following files:

- iosfwd.h
- streambuf_iterator.h

6.631 __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator Struct Reference

`#include <pat_trie_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator:



Public Types

- typedef _Alloc::difference_type **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef node_pointer_pointer **pointer**
- typedef node_pointer_reference **reference**
- typedef node_pointer **value_type**

Public Member Functions

- **iterator** (node_pointer_pointer p_p_cur=0, node_pointer_pointer p_p_end=0)
- bool **operator!=** (const const_iterator &other) const
- bool **operator!=** (const iterator &other) const
- node_const_pointer **operator*** () const
- node_pointer **operator*** ()
- iterator & **operator++** ()
- iterator **operator++** (int)
- const node_pointer_pointer **operator->** () const

- node_pointer_pointer **operator->** ()
- bool **operator==** (const const_iterator &other) const
- bool **operator==** (const iterator &other) const

Public Attributes

- node_pointer_pointer **m_p_p_cur**
- node_pointer_pointer **m_p_p_end**

6.631.1 Detailed Description

template<typename _ATraits, typename Metadata>
struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator

Child iterator.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

6.632 std::experimental::filesystem::v1::path::iterator Class Reference

`#include <fs_path.h>`

Public Types

- using **difference_type**
- using **iterator_category**
- using **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- **iterator** (const iterator &)=default
- reference **operator*** () const noexcept
- iterator & **operator++** () noexcept
- iterator **operator++** (int) noexcept
- iterator & **operator--** () noexcept
- iterator **operator--** (int) noexcept
- pointer **operator->** () const noexcept
- iterator & **operator=** (const iterator &)=default

Friends

- bool **operator!=** (const iterator &__lhs, const iterator &__rhs) noexcept
- bool **operator==** (const iterator &__lhs, const iterator &__rhs) noexcept
- class **path**

6.632.1 Detailed Description

An iterator for the components of a path.
The documentation for this class was generated from the following file:

- experimental/bits/fs_path.h

6.633 std::filesystem::path::iterator Class Reference

`#include <filesystem>`

Public Types

- using **difference_type**
- using **iterator_category**
- using **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- **iterator** (const iterator &)=default
- reference **operator*** () const noexcept
- iterator & **operator++** () noexcept
- iterator **operator++** (int) noexcept
- iterator & **operator--** () noexcept
- iterator **operator--** (int) noexcept
- pointer **operator->** () const noexcept
- iterator & **operator=** (const iterator &)=default

Friends

- void **__path_iter_advance** (iterator &__i, difference_type __n) noexcept
- difference_type **__path_iter_distance** (const iterator &__first, const iterator &__last) noexcept
- bool **operator!=** (const iterator &__lhs, const iterator &__rhs) noexcept
- bool **operator==** (const iterator &__lhs, const iterator &__rhs) noexcept
- class **path**

6.633.1 Detailed Description

An iterator for the components of a path.

Since

C++17

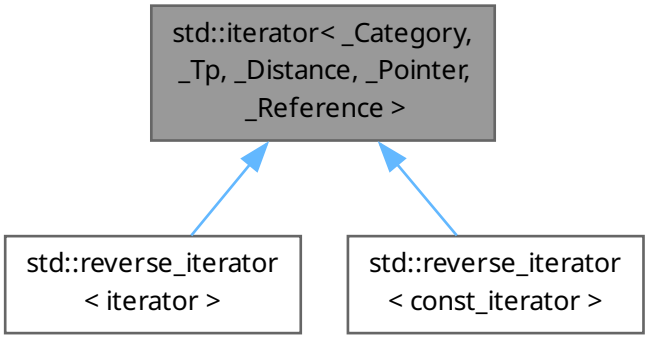
The documentation for this class was generated from the following file:

- bits/fs_path.h

6.634 std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference > Struct Template Reference

`#include <stl_iterator_base_types.h>`

Inheritance diagram for std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >:



Public Types

- typedef _Distance difference_type
- typedef _Category iterator_category
- typedef _Pointer pointer
- typedef _Reference reference
- typedef _Tp value_type

6.634.1 Detailed Description

template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _Tp*, typename _Reference = _Tp&>
struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >

Common iterator class.
This class does nothing but define nested typedefs. Iterator classes can inherit from this class to save some work. The typedefs are then used in specializations and overloading.
In particular, there are no default implementations of requirements such as `operator++` and the like. (How could there be?)

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

6.634.2 Member Typedef Documentation

difference_type

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↩
Tp*, typename _Reference = _Tp&>
typedef _Distance std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::difference_↩
type
```

Distance between iterators is represented as this type.

iterator_category

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↩
Tp*, typename _Reference = _Tp&>
typedef _Category std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::iterator_↩
category
```

One of the tag types.

pointer

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↩
Tp*, typename _Reference = _Tp&>
typedef _Pointer std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::pointer
```

This type represents a pointer-to-value_type.

reference

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↩
Tp*, typename _Reference = _Tp&>
typedef _Reference std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::reference
```

This type represents a reference-to-value_type.

value_type

```
template<typename _Category, typename _Tp, typename _Distance = ptrdiff_t, typename _Pointer = _↩
Tp*, typename _Reference = _Tp&>
typedef _Tp std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >::value_type
```

The type "pointed to" by the iterator.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

6.635 std::iterator_traits< _Iterator > Struct Template Reference

6.635.1 Detailed Description

template<typename _Iterator>
struct std::iterator_traits< _Iterator >

Traits class for iterators.
This class does nothing but define nested typedefs. The general version simply declares aliases for the nested typedefs from the Iterator argument. Partial specializations for pointers define the typedefs appropriately for the semantics of pointers.
The documentation for this struct was generated from the following file:

- iterator_concepts.h

6.636 std::iterator_traits< _Tp * > Struct Template Reference

```
#include <stl_iterator_base_types.h>
```

Public Types

- using **difference_type**
- using **iterator_category**
- using **iterator_concept**
- using **pointer**

- using **reference**
- using **value_type**

6.636.1 Detailed Description

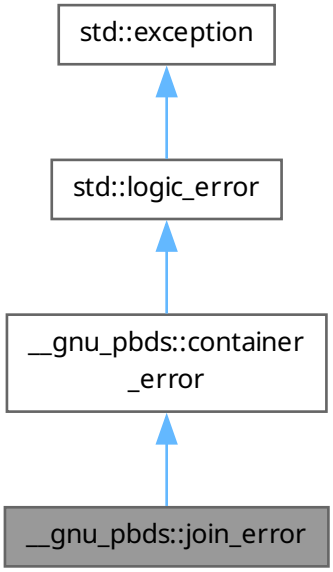
template<**typename _Tp**>
requires is_object_v<**_Tp**>
struct std::iterator_traits< **_Tp** * >

Partial specialization for object pointer types.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

6.637 __gnu_pbds::join_error Struct Reference

`#include <exception.hpp>`
Inheritance diagram for __gnu_pbds::join_error:



Public Member Functions

- virtual const char * what () const noexcept

6.637.1 Detailed Description

A join cannot be performed logical reasons (i.e., the ranges of the two container objects being joined overlaps.

6.637.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

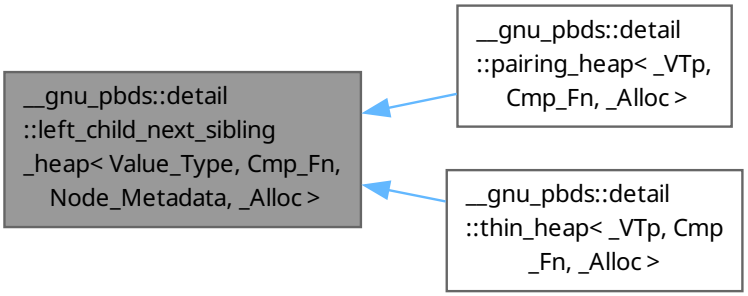
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
References what().
Referenced by what().
The documentation for this struct was generated from the following file:

- exception.hpp

6.638 __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > Class Template Reference

```
#include <left_child_next_sibling_heap_.hpp>
```

Inheritance diagram for __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _↩ Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef left_child_next_sibling_heap_const_iterator_< node, _Alloc > **const_iterator**
- typedef __rebind_v::const_pointer **const_pointer**
- typedef __rebind_v::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef const_iterator **iterator**
- typedef left_child_next_sibling_heap_node_< Value_Type, Node_Metadata, _Alloc > **node**
- typedef left_child_next_sibling_heap_node_point_const_iterator_< node, _Alloc > **point_const_iterator**
- typedef point_const_iterator **point_iterator**
- typedef __rebind_v::pointer **pointer**
- typedef __rebind_v::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **left_child_next_sibling_heap** (const Cmp_Fn &)
- **left_child_next_sibling_heap** (const left_child_next_sibling_heap &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- size_type **max_size** () const
- size_type **size** () const
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc > &)

Protected Types

- typedef alloc_traits::allocator_type **node_allocator**
- typedef alloc_traits::const_pointer **node_const_pointer**
- typedef Node_Metadata **node_metadata**
- typedef alloc_traits::pointer **node_pointer**
- typedef std::pair< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- node_pointer **get_new_node_for_insert** (const_reference)
- template<typename Pred>
  node_pointer **prune** (Pred)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- size_type **m_size**

6.638.1 Detailed Description

template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >

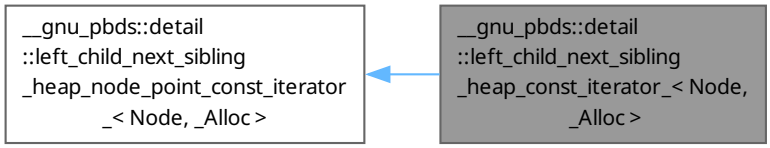
Base class for a basic heap.
The documentation for this class was generated from the following file:

- left_child_next_sibling_heap_.hpp

6.639 __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc > Class Template Reference

`#include <const_iterator.hpp>`

Inheritance diagram for __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >:



Public Types

- typedef base_type::const_pointer const_pointer
- typedef base_type::const_reference const_reference
- typedef _Alloc::difference_type difference_type
- typedef std::forward_iterator_tag iterator_category
- typedef base_type::pointer pointer
- typedef base_type::reference reference
- typedef base_type::value_type value_type

Public Member Functions

- left_child_next_sibling_heap_const_iterator_ ()
- left_child_next_sibling_heap_const_iterator_ (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other)
- **left_child_next_sibling_heap_const_iterator_** (node_pointer p_nd)
- bool operator!= (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other) const
- bool operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const
- const_reference operator* () const
- left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & **operator++** ()
- left_child_next_sibling_heap_const_iterator_< Node, _Alloc > **operator++** (int)
- const_pointer operator-> () const
- bool operator== (const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > &other) const
- bool operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

6.639.1 Detailed Description

template<typename Node, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >

Const point-type iterator.

6.639.2 Member Typedef Documentation

const_pointer

```
template<typename Node, typename _Alloc>
typedef base_type::const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::const_pointer
```

Iterator's const pointer type.

const_reference

```
template<typename Node, typename _Alloc>
typedef base_type::const_reference __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::const_reference
```

Iterator's const reference type.

difference_type

```
template<typename Node, typename _Alloc>
typedef _Alloc::difference_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::difference_type
```

Difference type.

iterator_category

```
template<typename Node, typename _Alloc>
typedef std::forward_iterator_tag __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::iterator_category
```

Category.

pointer

```
template<typename Node, typename _Alloc>
typedef base_type::pointer __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::pointer
```

Iterator's pointer type.

reference

```
template<typename Node, typename _Alloc>
typedef base_type::reference __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::reference
```

Iterator's reference type.

value_type

```
template<typename Node, typename _Alloc>
typedef base_type::value_type __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_<
Node, _Alloc >::value_type
```

Iterator's value type.

6.639.3 Constructor & Destructor Documentation

left_child_next_sibling_heap_const_iterator_() [1/2]

```
template<typename Node, typename _Alloc>
```

__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_↩
next_sibling_heap_const_iterator_ () [inline]
Default constructor.

left_child_next_sibling_heap_const_iterator_() [2/2]

```
template<typename Node, typename _Alloc>
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::left_child_↩
next_sibling_heap_const_iterator_ (
            const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other)  [inline]
```

Copy constructor.

6.639.4 Member Function Documentation

operator"!=() [1/2]

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator!=
(
            const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) const
[inline]
```

Compares content (negatively) to a different iterator object.

operator"!=() [2/2]

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator!= (
            const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other) const  [inline], [inherited]
```

Compares content (negatively) to a different iterator object.

operator*()

```
template<typename Node, typename _Alloc>
const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::operator* () const  [inline], [inherited]
```

Access.

operator->()

```
template<typename Node, typename _Alloc>
const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node,
_Alloc >::operator-> () const  [inline], [inherited]
```

Access.

operator==() [1/2]

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >::operator==
(
            const left_child_next_sibling_heap_const_iterator_< Node, _Alloc > & other) const
[inline]
```

Compares content to a different iterator object.

operator==() [2/2]

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator== (
            const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other) const  [inline], [inherited]
```

Compares content to a different iterator object.
The documentation for this class was generated from the following file:

- left_child_next_sibling_heap_/const_iterator.hpp

6.640 __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc > Struct Template Reference

`#include <node.hpp>`

Public Types

- typedef _Metadata **metadata_type**
- typedef rebind_traits< _Alloc, this_type >::pointer **node_pointer**
- typedef _Alloc::size_type **size_type**
- typedef _Value **value_type**

Public Attributes

- metadata_type **m_metadata**
- node_pointer **m_p_l_child**
- node_pointer **m_p_next_sibling**
- node_pointer **m_p_prev_or_parent**
- value_type **m_value**

6.640.1 Detailed Description

template<typename _Value, typename _Metadata, typename _Alloc>
struct __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc >

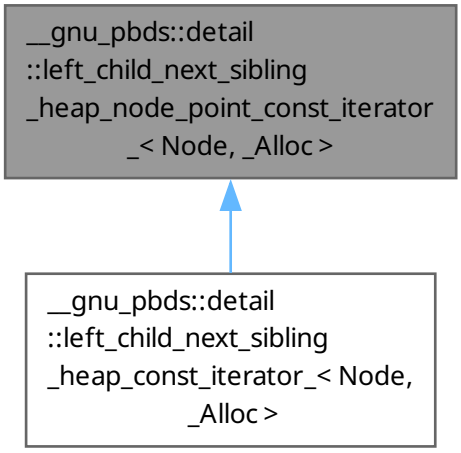
Node.
The documentation for this struct was generated from the following file:

- left_child_next_sibling_heap_/node.hpp

6.641 __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > Class Template Reference

`#include <point_const_iterator.hpp>`
Inheritance diagram for __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc

>:



Public Types

- typedef rebind_traits< _Alloc, value_type >::const_pointer const_pointer
- typedef rebind_traits< _Alloc, value_type >::const_reference const_reference
- typedef trivial_iterator_difference_type difference_type
- typedef trivial_iterator_tag iterator_category
- typedef rebind_traits< _Alloc, value_type >::pointer pointer
- typedef rebind_traits< _Alloc, value_type >::reference reference
- typedef Node::value_type value_type

Public Member Functions

- left_child_next_sibling_heap_node_point_const_iterator_ ()
- left_child_next_sibling_heap_node_point_const_iterator_ (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other)
- **left_child_next_sibling_heap_node_point_const_iterator_** (node_pointer p_nd)
- bool operator!= (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const
- const_reference operator* () const
- const_pointer operator-> () const
- bool operator== (const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &other) const

Public Attributes

- node_pointer **m_p_nd**

Protected Types

- typedef rebind_traits< _Alloc, Node >::pointer **node_pointer**

6.641.1 Detailed Description

template<typename Node, typename _Alloc>
class __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >

Const point-type iterator.

6.641.2 Member Typedef Documentation

const_pointer

```
template<typename Node, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_poi
Node, _Alloc >::const_pointer
```

Iterator's const pointer type.

const_reference

```
template<typename Node, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_p
Node, _Alloc >::const_reference
```

Iterator's const reference type.

difference_type

```
template<typename Node, typename _Alloc>
typedef trivial_iterator_difference_type __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterat
Node, _Alloc >::difference_type
```

Difference type.

iterator_category

```
template<typename Node, typename _Alloc>
typedef trivial_iterator_tag __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::iterator_category
```

Category.

pointer

```
template<typename Node, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_co
Node, _Alloc >::pointer
```

Iterator's pointer type.

reference

```
template<typename Node, typename _Alloc>
typedef rebind_traits<_Alloc,value_type>::reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_
Node, _Alloc >::reference
```

Iterator's reference type.

value_type

```
template<typename Node, typename _Alloc>
typedef Node::value_type __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::value_type
```

Iterator's value type.

6.641.3 Constructor & Destructor Documentation

left_child_next_sibling_heap_node_point_const_iterator_() [1/2]

```
template<typename Node, typename _Alloc>
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::↩
left_child_next_sibling_heap_node_point_const_iterator_ ()  [inline]
```

Default constructor.

left_child_next_sibling_heap_node_point_const_iterator_() [2/2]

```
template<typename Node, typename _Alloc>
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >::↩
left_child_next_sibling_heap_node_point_const_iterator_ (
            const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other)  [inline]
```

Copy constructor.

6.641.4 Member Function Documentation

operator"!=()

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator!= (
            const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other) const  [inline]
```

Compares content (negatively) to a different iterator object.

operator∗()

```
template<typename Node, typename _Alloc>
const_reference __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_<
Node, _Alloc >::operator* () const  [inline]
```

Access.

operator->()

```
template<typename Node, typename _Alloc>
const_pointer __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node,
_Alloc >::operator-> () const  [inline]
```

Access.

operator==()

```
template<typename Node, typename _Alloc>
bool __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc
>::operator== (
            const left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc > &
other) const  [inline]
```

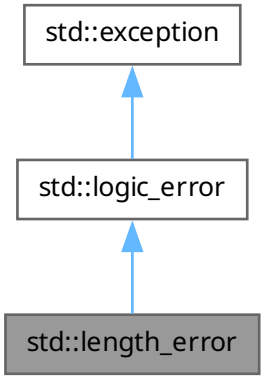
Compares content to a different iterator object.

The documentation for this class was generated from the following file:

- left_child_next_sibling_heap_/point_const_iterator.hpp

6.642 std::length_error Class Reference

`#include <stdexcept_except.h>`
Inheritance diagram for std::length_error:



Public Member Functions

- **length_error** (const char *)
- **length_error** (const length_error &)=default
- **length_error** (const string &__arg)
- **length_error** (length_error &&)=default
- length_error & **operator=** (const length_error &)=default
- length_error & **operator=** (length_error &&)=default
- virtual const char * what () const noexcept

6.642.1 Detailed Description

Thrown when an object is constructed that would exceed its maximum permitted size (e.g., a basic_string instance).

6.642.2 Member Function Documentation

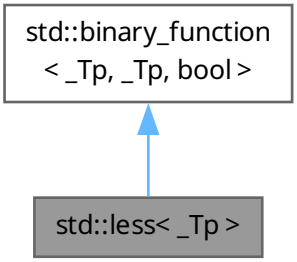
what()

`virtual const char * std::logic_error::what () const [virtual], [noexcept], [inherited]`
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
References what().
Referenced by what().
The documentation for this class was generated from the following file:

- stdexcept_except.h

6.643 std::less< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::less< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

6.643.1 Detailed Description

template<typename _Tp>
struct std::less< _Tp >

One of the comparison functors.

6.643.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.644 std::ranges::less Struct Reference

`#include <ranges_cmp.h>`

Public Types

- using **is_transparent**

Static Public Member Functions

- template<typename _Tp, typename _Up>
  static constexpr bool **operator()** (_Tp &&__t, _Up &&__u) noexcept(noexcept(std::declval< _Tp >()< std::declval< _Up >()))

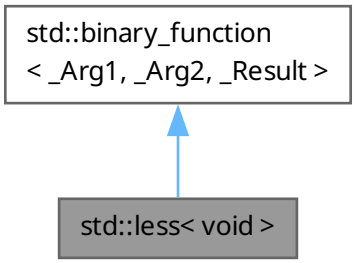
6.644.1 Detailed Description

ranges::less function object type.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

6.645 std::less< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::less< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t)< std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t)< std::forward< _Up >(__u))
- template<typename _Tp, typename _Up>
  constexpr bool **operator()** (_Tp *__t, _Up *__u) const noexcept

6.645.1 Detailed Description

One of the comparison functors.

6.645.2 Member Typedef Documentation

first_argument_type

```
typedef void std::binary_function< void, void, bool >::first_argument_type
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< void, void, bool >::result_type
```

`result_type` is the return type

second_argument_type

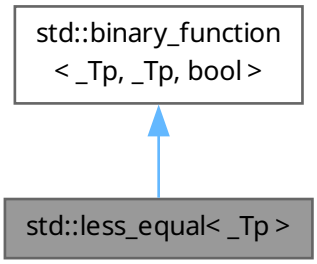
```
typedef void std::binary_function< void, void, bool >::second_argument_type
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.646 std::less_equal< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::less_equal< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

6.646.1 Detailed Description

template<typename _Tp>
struct std::less_equal< _Tp >

One of the comparison functors.

6.646.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.647 std::ranges::less_equal Struct Reference

```
#include <ranges_cmp.h>
```

Public Types

- using **is_transparent**

Static Public Member Functions

- template<typename _Tp, typename _Up>
  static constexpr bool **operator()** (_Tp &&__t, _Up &&__u) noexcept(noexcept(std::declval< _Up >()< std::declval< _Tp >()))

6.647.1 Detailed Description

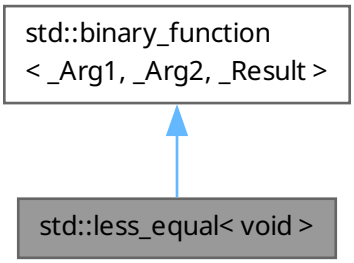
ranges::less_equal function object type.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

6.648 std::less_equal< void > Struct Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::less_equal< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t)<=std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t)<=std::forward< _Up >(__u))
- template<typename _Tp, typename _Up>
  constexpr bool **operator()** (_Tp *__t, _Up *__u) const noexcept

6.648.1 Detailed Description

One of the comparison functors.

6.648.2 Member Typedef Documentation

first_argument_type

```
typedef void std::binary_function< void, void, bool >::first_argument_type
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< void, void, bool >::result_type
```

`result_type` is the return type

second_argument_type

```
typedef void std::binary_function< void, void, bool >::second_argument_type
```

`second_argument_type` is the type of the second argument

The documentation for this struct was generated from the following file:

- stl_function.h

6.649 __gnu_cxx::limit_condition::limit_adjustor Struct Reference

`#include <throw_allocator.h>`

Public Member Functions

- **limit_adjustor** (const size_t __l)

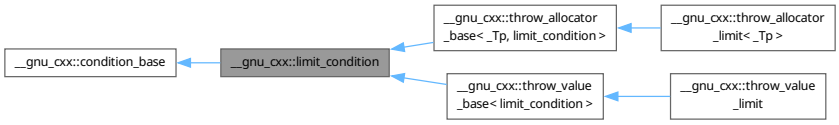
6.649.1 Detailed Description

Enter the nth condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.650 __gnu_cxx::limit_condition Struct Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::limit_condition:



Classes

- struct always_adjustor
- struct limit_adjustor
- struct never_adjustor

Static Public Member Functions

- static size_t & **count** ()
- static size_t & **limit** ()
- static void **set_limit** (const size_t __l)
- static void **throw_conditionally** ()

6.650.1 Detailed Description

Base class for incremental control and throw.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.651 std::linear_congruential_engine< _UIntType, __a, __c, __m > Class Template Reference

`#include <random>`

Public Types

- typedef _UIntType result_type

Public Member Functions

- linear_congruential_engine ()
- template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
  linear_congruential_engine (_Sseq &__q)
- linear_congruential_engine (result_type __s)
- void discard (unsigned long long __z)
- result_type operator() ()
- template<typename _Sseq>
  _If_seed_seq< _Sseq > seed (_Sseq &__q)
- void seed (result_type __s=default_seed)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr result_type **default_seed**
- static constexpr result_type increment
- static constexpr result_type modulus
- static constexpr result_type multiplier

Friends

- template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > &__lcr)
- bool operator== (const linear_congruential_engine &__lhs, const linear_congruential_engine &__rhs)
- template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > &__lcr)

6.651.1 Detailed Description

template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
class std::linear_congruential_engine< _UIntType, __a, __c, __m >

A model of a linear congruential random number generator.
A random number generator that produces pseudorandom numbers via linear function:

$$x_{i+1} \leftarrow (ax_i + c) \bmod m$$

The template parameter `_UIntType` must be an unsigned integral type large enough to store values up to (__m-1). If the template parameter `__m` is 0, the modulus `__m` used is std::numeric_limits<_UIntType>::max() plus 1. Otherwise, the template parameters `__a` and `__c` must be less than `__m`.
The size of the state is 1.

Since

C++11

6.651.2 Member Typedef Documentation

result_type

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
typedef _UIntType std::linear_congruential_engine< _UIntType, __a, __c, __m >::result_type
```

The type of the generated random value.

6.651.3 Constructor & Destructor Documentation

linear_congruential_engine() [1/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine ()  [inline]
```

Constructs a linear_congruential_engine random number generator engine with seed 1.

linear_congruential_engine() [2/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (
            result_type __s)  [inline], [explicit]
```

Constructs a linear_congruential_engine random number generator engine with seed __s. The default seed value is 1.

Parameters

_↩ _s	The initial seed value.

linear_congruential_engine() [3/3]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::linear_congruential_engine< _UIntType, __a, __c, __m >::linear_congruential_engine (
            _Sseq & __q)  [inline], [explicit]
```

Constructs a linear_congruential_engine random number generator engine seeded from the seed sequence __q.

Parameters

_↩ _q	the seed sequence.

6.651.4 Member Function Documentation

discard()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
void std::linear_congruential_engine< _UIntType, __a, __c, __m >::discard (
            unsigned long long __z)  [inline]
```

Discard a sequence of random numbers.

max()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
constexpr result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::max ()  [inline],
```

[static], [constexpr]
Gets the largest possible value in the output range.

min()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
constexpr result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::min ()  [inline],
[static], [constexpr]
```
Gets the smallest possible value in the output range.
The minimum depends on the __c parameter: if it is zero, the minimum generated must be > 0, otherwise 0 is allowed.

operator()()

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::operator() ()  [inline]
```
Gets the next random number in the sequence.

seed() [1/2]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _Sseq>
auto std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
            _Sseq & __q) -> _If_seed_seq< _Sseq >
```
Reseeds the linear_congruential_engine random number generator engine sequence using values from the seed sequence __q.

Parameters

_↩ _q	the seed sequence.

Seeds the LCR engine with a value generated by __q.
References std::__lg(), std::__numeric_limits_base::digits, and seed().

seed() [2/2]

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
void std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed (
            result_type __s = default_seed)
```
Reseeds the linear_congruential_engine random number generator engine sequence to the seed __s.

Parameters

_↩ _s	The new seed.

Seeds the LCR with integral value __s, adjusted so that the ring identity is never a member of the convergence set.
Referenced by std::linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >::linear_congruential_engine(), std::linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL >::linear_congruential_engine(), and seed().

6.651.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT,
typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > & __lcr)  [friend]
```

Writes the textual representation of the state x(i) of x to __os.

Parameters

__os	The output stream.
__lcr	A % linear_congruential_engine random number generator.

Returns

__os.

operator==

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
bool operator== (
            const linear_congruential_engine< _UIntType, __a, __c, __m > & __lhs,
            const linear_congruential_engine< _UIntType, __a, __c, __m > & __rhs)  [friend]
```

Compares two linear congruential random number generator objects of the same type for equality.

Parameters

__lhs	A linear congruential random number generator object.
__rhs	Another linear congruential random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

operator>>

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
template<typename _UIntType1, _UIntType1 __a1, _UIntType1 __c1, _UIntType1 __m1, typename _CharT,
typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::linear_congruential_engine< _UIntType1, __a1, __c1, __m1 > & __lcr)  [friend]
```

Sets the state of the engine by reading its textual representation from __is.
The textual representation must have been previously written using an output stream whose imbued locale and whose type's template specialization arguments _CharT and _Traits were the same as those of __is.

Parameters

__is	The input stream.
__lcr	A % linear_congruential_engine random number generator.

Returns

__is.

6.651.6 Member Data Documentation

increment

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::increment  [static],
[constexpr]
```

An increment.

modulus

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::modulus  [static], [constexpr]
```

The modulus.

multiplier

```
template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m>
result_type std::linear_congruential_engine< _UIntType, __a, __c, __m >::multiplier  [static],
[constexpr]
```

The multiplier.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.652 __gnu_pbds::linear_probe_fn< Size_Type > Class Template Reference

```
#include <hash_policy.hpp>
```

Public Types

- typedef Size_Type **size_type**

Public Member Functions

- void **swap** (linear_probe_fn< Size_Type > &other)

Protected Member Functions

- size_type operator() (size_type i) const

6.652.1 Detailed Description

template<typename Size_Type = std::size_t>
class __gnu_pbds::linear_probe_fn< Size_Type >

A probe sequence policy using fixed increments.

6.652.2 Member Function Documentation

operator()()

```
template<typename Size_Type>
linear_probe_fn< Size_Type >::size_type __gnu_pbds::linear_probe_fn< Size_Type >::operator() (
            size_type i) const  [inline], [protected]
```

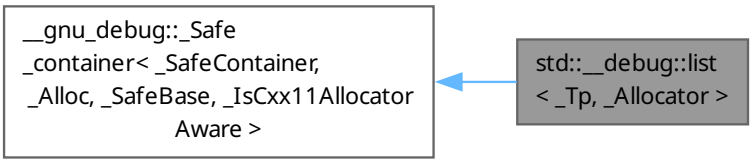
Returns the i-th offset from the hash value.
The documentation for this class was generated from the following file:

- hash_policy.hpp

6.653 std::__debug::list< _Tp, _Allocator > Class Template Reference

```
#include <list>
```

Inheritance diagram for std::__debug::list< _Tp, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, list > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, list > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **list** (_Base_ref __x)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  list (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- **list** (const _Allocator &__a) noexcept
- **list** (const list &)=default
- **list** (const list &__x, const __type_identity_t< allocator_type > &__a)
- **list** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- **list** (list &&)=default
- **list** (list &&__x, const __type_identity_t< allocator_type > &__a) noexcept(std::is_nothrow_constructible< _Base, _Base, const allocator_type & >::value)
- **list** (size_type __n, const __type_identity_t< _Tp > &__value, const _Allocator &__a=_Allocator())
- **list** (size_type __n, const allocator_type &__a=allocator_type())
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (initializer_list< value_type > __l)
- void **assign** (size_type __n, const _Tp &__t)
- const_reference **back** () const noexcept
- reference **back** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (const_iterator __position, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- iterator **erase** (const_iterator __first, const_iterator __last) noexcept
- iterator **erase** (const_iterator __position) noexcept
- const_reference **front** () const noexcept
- reference **front** () noexcept
- iterator **insert** (const_iterator __p, initializer_list< value_type > __l)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator **insert** (const_iterator __position, _InputIterator __first, _InputIterator __last)
- iterator **insert** (const_iterator __position, _Tp &&__x)
- iterator **insert** (const_iterator __position, const _Tp &__x)
- iterator **insert** (const_iterator __position, size_type __n, const _Tp &__x)
- void **merge** (list &&__x)
- template<class _Compare>
  void **merge** (list &&__x, _Compare __comp)
- void **merge** (list &__x)
- template<typename _Compare>
  void **merge** (list &__x, _Compare __comp)
- list & **operator=** (const list &)=default
- list & **operator=** (initializer_list< value_type > __l)
- list & **operator=** (list &&)=default

- void **pop_back** () noexcept
- void **pop_front** () noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- __remove_return_type **remove** (const _Tp &__value)
- template<class _Predicate>
  __remove_return_type **remove_if** (_Predicate __pred)
- const_reverse_iterator **rend** () const noexcept
- reverse_iterator **rend** () noexcept
- void **resize** (size_type __sz)
- void **resize** (size_type __sz, const _Tp &__c)
- void **sort** ()
- template<typename _StrictWeakOrdering>
  void **sort** (_StrictWeakOrdering __pred)
- void **splice** (const_iterator __position, list &&__x) noexcept
- void **splice** (const_iterator __position, list &&__x, const_iterator __first, const_iterator __last) noexcept
- void **splice** (const_iterator __position, list &&__x, const_iterator __i) noexcept
- void **splice** (const_iterator __position, list &__x) noexcept
- void **splice** (const_iterator __position, list &__x, const_iterator __first, const_iterator __last) noexcept
- void **splice** (const_iterator __position, list &__x, const_iterator __i) noexcept
- void **swap** (list &__x) noexcept(/*conditional */)
- __remove_return_type **unique** ()
- template<class _BinaryPredicate>
  __remove_return_type **unique** (_BinaryPredicate __binary_pred)

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

6.653.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__debug::list< _Tp, _Allocator >

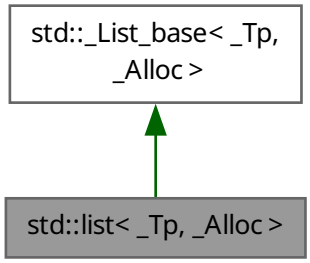
Class std::list with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/list

6.654 std::list< _Tp, _Alloc > Class Template Reference

`#include <stl_list.h>`

Inheritance diagram for std::list< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _Node_traits::_Const_iterator **const_iterator**
- typedef _Tp_alloc_traits::const_pointer **const_pointer**
- typedef _Tp_alloc_traits::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef _Node_traits::_Iterator **iterator**
- typedef _Tp_alloc_traits::pointer **pointer**
- typedef _Tp_alloc_traits::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- list ()=default
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  list (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- list (const allocator_type &__a) noexcept
- list (const list &__x)
- **list** (const list &__x, const __type_identity_t< allocator_type > &__a)
- list (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- list (list &&)=default
- **list** (list &&__x, const __type_identity_t< allocator_type > &__a) noexcept(_Node_alloc_traits::_S_always_↩
  equal())
- list (size_type __n, const allocator_type &__a=allocator_type())
- list (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- ~list ()=default
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  void assign (_InputIterator __first, _InputIterator __last)
- void assign (initializer_list< value_type > __l)
- void assign (size_type __n, const value_type &__val)

- const_reference back () const noexcept
- reference back () noexcept
- const_iterator begin () const noexcept
- iterator begin () noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- template<typename... _Args>
  iterator emplace (const_iterator __position, _Args &&... __args)
- template<typename... _Args>
  reference **emplace_back** (_Args &&... __args)
- template<typename... _Args>
  reference **emplace_front** (_Args &&... __args)
- bool empty () const noexcept
- const_iterator end () const noexcept
- iterator end () noexcept
- iterator erase (const_iterator __first, const_iterator __last) noexcept
- iterator erase (const_iterator __position) noexcept
- const_reference front () const noexcept
- reference front () noexcept
- allocator_type get_allocator () const noexcept
- iterator insert (const_iterator __p, initializer_list< value_type > __l)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)
- iterator insert (const_iterator __position, size_type __n, const value_type &__x)
- size_type max_size () const noexcept
- void merge (list &&__x)
- template<typename _StrictWeakOrdering>
  void merge (list &&__x, _StrictWeakOrdering __comp)
- void **merge** (list &__x)
- template<typename _StrictWeakOrdering>
  void **merge** (list &__x, _StrictWeakOrdering __comp)
- list & operator= (const list &__x)
- list & operator= (initializer_list< value_type > __l)
- list & operator= (list &&__x) noexcept(_Node_alloc_traits::_S_nothrow_move())
- void pop_back () noexcept
- void pop_front () noexcept
- void push_back (const value_type &__x)
- void **push_back** (value_type &&__x)
- void push_front (const value_type &__x)
- void **push_front** (value_type &&__x)
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rbegin () noexcept
- __remove_return_type remove (const _Tp &__value)
- template<typename _Predicate>
  __remove_return_type remove_if (_Predicate)
- const_reverse_iterator rend () const noexcept
- reverse_iterator rend () noexcept
- void resize (size_type __new_size)

- void resize (size_type __new_size, const value_type &__x)
- void reverse () noexcept
- size_type size () const noexcept
- void sort ()
- template<typename _StrictWeakOrdering>
  void sort (_StrictWeakOrdering)
- void splice (const_iterator __position, list &&__x) noexcept
- void splice (const_iterator __position, list &&__x, const_iterator __first, const_iterator __last) noexcept
- void **splice** (const_iterator __position, list &__x) noexcept
- void splice (const_iterator __position, list &__x, const_iterator __first, const_iterator __last) noexcept
- void swap (list &__x) noexcept
- __remove_return_type unique ()
- template<typename _BinaryPredicate>
  __remove_return_type unique (_BinaryPredicate)

- iterator insert (const_iterator __position, const value_type &__x)
- iterator insert (const_iterator __position, value_type &&__x)

- void splice (const_iterator __position, list &&__x, const_iterator __i) noexcept
- void splice (const_iterator __position, list &__x, const_iterator __i) noexcept

Protected Types

- typedef _Node_alloc_traits::pointer **_Node_ptr**

Protected Member Functions

- template<typename _InputIterator>
  void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename _Integer>
  void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- void **_M_check_equal_allocators** (const list &__x) noexcept
- void **_M_clear** () noexcept
- template<typename... _Args>
  _Node_ptr _M_create_node (_Args &&... __args)
- void **_M_dec_size** (size_t __n)
- void **_M_default_append** (size_type __n)
- void **_M_default_initialize** (size_type __n)
- void **_M_destroy_node** (_Node_ptr __p)
- size_t **_M_distance** (const __detail::_List_node_base *__first, const __detail::_List_node_base *__last) const
- void **_M_erase** (iterator __position) noexcept
- void **_M_fill_assign** (size_type __n, const value_type &__val)
- void **_M_fill_initialize** (size_type __n, const value_type &__x)
- _Node_alloc_traits::pointer **_M_get_node** ()
- const _Node_alloc_type & **_M_get_Node_allocator** () const noexcept
- _Node_alloc_type & **_M_get_Node_allocator** () noexcept
- size_t **_M_get_size** () const
- void **_M_inc_size** (size_t __n)
- void **_M_init** () noexcept

- template<typename _InputIterator>
  void **_M_initialize_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename _Integer>
  void **_M_initialize_dispatch** (_Integer __n, _Integer __x, __true_type)
- template<typename... _Args>
  void **_M_insert** (iterator __position, _Args &&... __args)
- void **_M_move_assign** (list &&__x, false_type)
- void **_M_move_assign** (list &&__x, true_type) noexcept
- void **_M_move_nodes** (_List_base &&__x)
- size_t **_M_node_count** () const
- void **_M_put_node** (_Node_ptr __p) noexcept
- const_iterator **_M_resize_pos** (size_type &__new_size) const
- void **_M_set_size** (size_t __n)
- void **_M_transfer** (iterator __position, iterator __first, iterator __last)

Static Protected Member Functions

- static size_t **_S_distance** (const __detail::_List_node_base *__first, const __detail::_List_node_base *__last)
- static size_t **_S_distance** (const_iterator __first, const_iterator __last)

Protected Attributes

- _List_impl **_M_impl**

6.654.1 Detailed Description

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::list< _Tp, _Alloc >

A standard container with linear time access to elements, and fixed time insertion/deletion at any point in the sequence.

Template Parameters

_Tp	Type of element.
_Alloc	Allocator type, defaults to allocator<_Tp>.

Meets the requirements of a `container`, a `reversible container`, and a `sequence`, including the `optional sequence requirements` with the exception of `at` and `operator[]`.
This is a *doubly linked* list. Traversal up and down the list requires linear time, but adding and removing elements (or *nodes*) is done in constant time, regardless of where the change takes place. Unlike std::vector and std::deque, random-access iterators are not provided, so subscripting ([]) access is not allowed. For algorithms which only need sequential access, this lack makes no difference.
Also unlike the other standard containers, std::list provides specialized algorithms unique to linked lists, such as splicing, sorting, and in-place reversal.
A couple points on memory allocation for list<Tp>:
First, we never actually allocate a Tp, we allocate List_node<Tp>'s and trust [20.1.5]/4 to DTRT. This is to ensure that after elements from list<X,Alloc1> are spliced into list<X,Alloc2>, destroying the memory of the second list is a valid operation, i.e., Alloc1 giveth and Alloc2 taketh away.
Second, a list conceptually represented as

```
A <---> B <---> C <---> D
```

is actually circular; a link exists between A and D. The list class holds (as its only data member) a private list::iterator pointing to *D*, not to *A!* To get to the head of the list, we start at the tail and move forward by one. When this member iterator's next/previous pointers refer to itself, the list is empty.

6.654.2 Constructor & Destructor Documentation

list() [1/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list ()  [default]
```

Creates a list with no elements.

Referenced by erase(), insert(), insert(), insert(), merge(), merge(), operator=(), remove(), remove_if(), unique(), and unique().

list() [2/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
            const allocator_type & __a)  [inline], [explicit], [noexcept]
```

Creates a list with no elements.

Parameters

_↩ _a	An allocator object.

list() [3/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
            size_type __n,
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Creates a list with default constructed elements.

Parameters

_↩ _n	The number of elements to initially create.
_↩ _a	An allocator object.

This constructor fills the list with __n default constructed elements.

list() [4/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
            size_type __n,
            const value_type & __value,
            const allocator_type & __a = allocator_type())  [inline]
```

Creates a list with copies of an exemplar element.

Parameters

__n	The number of elements to initially create.
__value	An element to copy.

__a	An allocator object.

This constructor fills the list with *__n* copies of *__value*.

list() [5/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
            const list< _Tp, _Alloc > & __x)  [inline]
```

List copy constructor.

Parameters

_↩ _x	A list of identical element and allocator types.

The newly-created list uses a copy of the allocation object used by *__x* (unless the allocator traits dictate a different object).

list() [6/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
            list< _Tp, _Alloc > && )  [default]
```

List move constructor.

The newly-created list contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified list.

list() [7/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::list (
            initializer_list< value_type > __l,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a list from an initializer_list.

Parameters

_↩ _l	An initializer_list of value_type.
_↩ _a	An allocator object.

Create a list consisting of copies of the elements in the initializer_list *__l*. This is linear in __l.size().

list() [8/8]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
```

```
std::list< _Tp, _Alloc >::list (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a list from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__a	An allocator object.

Create a list consisting of copies of the elements from [*__first*,*__last*). This is linear in N (where N is distance(*__first*,↩ *__last*)).

~list()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::list< _Tp, _Alloc >::~list ()  [default]
```

No explicit dtor needed as the _Base dtor takes care of things. The _Base dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

6.654.3 Member Function Documentation

_M_create_node()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename...  _Args>
_Node_ptr std::list< _Tp, _Alloc >::_M_create_node (
            _Args &&...  __args)  [inline], [protected]
```

Parameters

__args	An instance of user data.

Allocates space for a new node and constructs a copy of *__args* in it.
Referenced by emplace(), and insert().

assign() [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::list< _Tp, _Alloc >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Assigns a range to a list.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a list with copies of the elements in the range [__*first*,__*last*).
Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned.

assign() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::assign (
            initializer_list< value_type > __l)  [inline]
```

Assigns an initializer_list to a list.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list of value_type.

Replace the contents of the list with copies of the elements in the initializer_list __*l*. This is linear in __l.size().

assign() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::assign (
            size_type __n,
            const value_type & __val)  [inline]
```

Assigns a given value to a list.

Parameters

__*n*	Number of elements to be assigned.
__*val*	Value to be assigned.

This function fills a list with __*n* copies of the given value. Note that the assignment completely changes the list and that the resulting list's size is the same as the number of elements assigned.

back() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list< _Tp, _Alloc >::back () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the list.

back() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list< _Tp, _Alloc >::back ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the last element of the list.

begin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::begin () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

begin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::begin ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points to the first element in the list. Iteration is done in ordinary element order.
Referenced by std::list< _Tp, polymorphic_allocator< _Tp > >::list(), insert(), insert(), merge(), merge(), std::operator<=>(), operator=(), std::list< _Tp, polymorphic_allocator< _Tp > >::operator=(), remove(), remove_if(), std::list< _Tp, polymorphic_allocator< _ unique(), and unique().

cbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the list. Iteration is done in ordinary element order.

cend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

clear()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
Referenced by operator=().

crbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::crbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

crend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::crend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

emplace()

```
template<typename _Tp, typename _Alloc>
template<typename...  _Args>
list< _Tp, _Alloc >::iterator list::emplace (
            const_iterator __position,
            _Args &&...  __args)
```

Constructs object in list before specified iterator.

Parameters

__position	A const_iterator into the list.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.
References _M_create_node(), and std::forward().

empty()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::list< _Tp, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the list is empty. (Thus begin() would equal end().)
Referenced by insert(), and std::list< _Tp, polymorphic_allocator< _Tp > >::splice().

end() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::list< _Tp, _Alloc >::end () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the list. Iteration is done in ordinary element order.

end() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::end ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write iterator that points one past the last element in the list. Iteration is done in ordinary element order.
Referenced by std::list< _Tp, polymorphic_allocator< _Tp > >::list(), merge(), merge(), std::operator<=>(), operator=(), std::list< _Tp, polymorphic_allocator< _Tp > >::operator=(), and std::list< _Tp, polymorphic_allocator< _Tp > >::splice().

erase() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline], [noexcept]
```

Remove a range of elements.

Parameters

__first	Iterator pointing to the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *last* prior to erasing (or end()).

This function will erase the elements in the range [first,last) and shorten the list accordingly.
This operation is linear time in the size of the range and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/2]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::iterator list::erase (
            const_iterator __position)  [noexcept]
```

Remove element at given position.

Parameters

__position	Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the list by one.
Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References list(), and erase().
Referenced by erase(), resize(), and resize().

front() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::list< _Tp, _Alloc >::front () const  [inline], [nodiscard], [noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the list.

front() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::list< _Tp, _Alloc >::front ()  [inline], [nodiscard], [noexcept]
```

Returns a read/write reference to the data at the first element of the list.

get_allocator()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::list< _Tp, _Alloc >::get_allocator () const  [inline], [noexcept]
```

Get a copy of the memory allocation object.
Referenced by insert(), insert(), remove(), remove_if(), unique(), and unique().

insert() [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::insert (
            const_iterator __p,
            initializer_list< value_type > __l)  [inline]
```

Inserts the contents of an initializer_list into list before specified const_iterator.

Parameters

__p	A const_iterator into the list.
__l	An initializer_list of value_type.

Returns

An iterator pointing to the first element inserted (or __p).

This function will insert copies of the data in the initializer_list __l into the list before the location specified by __p.
This operation is linear in the number of elements inserted and does not invalidate iterators and references.

insert() [2/5]

```
template<typename _Tp, typename _Alloc>
template<typename _InputIterator, typename>
list< _Tp, _Alloc >::iterator list::insert (
            const_iterator __position,
            _InputIterator __first,
            _InputIterator __last)
```

Inserts a range into the list.

Parameters

__position	An iterator into the list.
__first	An input iterator.
__last	An input iterator.

Returns

Since C++11, an iterator pointing to the first element inserted (or __position). In C++98 this returns void.

This function will insert copies of the data in the range [__first,__last) into the list before the location specified by __position.
This operation is linear in the number of elements inserted and does not invalidate iterators and references.
References list(), begin(), empty(), get_allocator(), and splice().

insert() [3/5]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::iterator list::insert (
            const_iterator __position,
            const value_type & __x)
```

Inserts given value into list before specified iterator.

Parameters

__position	An iterator into the list.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.
References list(), _M_create_node(), and insert().
Referenced by insert(), and resize().

insert() [4/5]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::iterator list::insert (
             const_iterator __position,
             size_type __n,
             const value_type & __x)
```

Inserts a number of copies of given data into the list.

Parameters

__position	An iterator into the list.
__n	Number of elements to be inserted.
__x	Data to be inserted.

Returns

Since C++11, an iterator pointing to the first element inserted (or `__position`). In C++98 this returns void.

This function will insert a specified number of copies of the given data before the location specified by `__position`. This operation is linear in the number of elements inserted and does not invalidate iterators and references.
References list(), begin(), get_allocator(), and splice().

insert() [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::list< _Tp, _Alloc >::insert (
             const_iterator __position,
             value_type && __x)  [inline]
```

Inserts given value into list before specified iterator.

Parameters

__position	An iterator into the list.

__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

max_size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list< _Tp, _Alloc >::max_size () const  [inline], [nodiscard], [noexcept]
```

Returns the size() of the largest possible list.

merge() [1/2]

```
template<typename _Tp, typename _Alloc>
void list::merge (
            list< _Tp, _Alloc > && __x)
```

Merge sorted lists.

Parameters

_↩ _x	Sorted list to merge.

Assumes that both __x and this list are sorted according to operator<(). Merges elements of __x into this list in sorted order, leaving __x empty when complete. Elements in this list precede elements in __x that are equal.
References list(), std::__addressof(), std::begin(), begin(), std::end(), end(), and merge().
Referenced by merge(), and merge().

merge() [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _StrictWeakOrdering>
void list::merge (
            list< _Tp, _Alloc > && __x,
            _StrictWeakOrdering __comp)
```

Merge sorted lists according to comparison function.

Template Parameters

_StrictWeakOrdering	Comparison function defining sort order.

Parameters

__x	Sorted list to merge.
__comp	Comparison functor.

Assumes that both __x and this list are sorted according to StrictWeakOrdering. Merges elements of __x into this list in sorted order, leaving __x empty when complete. Elements in this list precede elements in __x that are equivalent according to StrictWeakOrdering().
References list(), std::__addressof(), std::begin(), begin(), std::end(), end(), and merge().

operator=() [1/3]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc > & list::operator= (
            const list< _Tp, _Alloc > & __x)
```

List assignment operator.

Parameters

_↩ _x	A list of identical element and allocator types.

All the elements of __x are copied.
Whether the allocator is copied depends on the allocator traits.
References list(), std::__addressof(), begin(), clear(), and end().

operator=() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list & std::list< _Tp, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

List initializer list assignment operator.

Parameters

↩ _↩ ↩ _↩ l	An initializer_list of value_type.

Replace the contents of the list with copies of the elements in the initializer_list __l. This is linear in l.size().

operator=() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
list & std::list< _Tp, _Alloc >::operator= (
            list< _Tp, _Alloc > && __x)  [inline], [noexcept]
```

List move assignment operator.

Parameters

_↩ _x	A list of identical element and allocator types.

The contents of __x are moved into this list (without copying).
Afterwards __x is a valid, but unspecified list
Whether the allocator is moved depends on the allocator traits.

pop_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::pop_back ()  [inline], [noexcept]
```

Removes last element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop_back() is called.

pop_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::pop_front ()  [inline], [noexcept]
```

Removes first element.

This is a typical stack operation. It shrinks the list by one. Due to the nature of a list this operation can be done in constant time, and only invalidates iterators/references to the element being removed.

Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop_front() is called.

push_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::push_back (
            const value_type & __x)  [inline]
```

Add data to the end of the list.

Parameters

_↩ _x	Data to be added.

This is a typical stack operation. The function creates an element at the end of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

push_front()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::push_front (
            const value_type & __x)  [inline]
```

Add data to the front of the list.

Parameters

_↩ _x	Data to be added.

This is a typical stack operation. The function creates an element at the front of the list and assigns the given data to it. Due to the nature of a list this operation can be done in constant time, and does not invalidate iterators and references.

rbegin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
```

```
const_reverse_iterator std::list< _Tp, _Alloc >::rbegin () const  [inline], [nodiscard], [noexcept]
```
Returns a read-only (constant) reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

rbegin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::list< _Tp, _Alloc >::rbegin ()  [inline], [nodiscard], [noexcept]
```
Returns a read/write reverse iterator that points to the last element in the list. Iteration is done in reverse element order.

remove()

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::__remove_return_type list::remove (
            const _Tp & __value)
```
Remove all elements equal to value.

Parameters

__value	The value to remove.

Removes every element in the list equal to *value*. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References list(), std::begin(), begin(), std::end(), get_allocator(), size(), and splice().

remove_if()

```
template<typename _Tp, typename _Alloc>
template<typename _Predicate>
list< _Tp, _Alloc >::__remove_return_type list::remove_if (
            _Predicate __pred)
```
Remove all elements satisfying a predicate.

Template Parameters

_Predicate	Unary predicate function or object.

Removes every element in the list for which the predicate returns true. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References list(), std::begin(), begin(), std::end(), get_allocator(), size(), and splice().

rend() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::list< _Tp, _Alloc >::rend () const  [inline], [nodiscard], [noexcept]
```
Returns a read-only (constant) reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

rend() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
```

```
reverse_iterator std::list< _Tp, _Alloc >::rend ()  [inline], [nodiscard], [noexcept]
```
Returns a read/write reverse iterator that points to one before the first element in the list. Iteration is done in reverse element order.

resize() [1/2]

```
template<typename _Tp, typename _Alloc>
void list::resize (
            size_type __new_size)
```
Resizes the list to the specified number of elements.

Parameters

__new_size	Number of elements the list should contain.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise default constructed elements are appended.
References std::end(), and erase().

resize() [2/2]

```
template<typename _Tp, typename _Alloc>
void list::resize (
            size_type __new_size,
            const value_type & __x)
```
Resizes the list to the specified number of elements.

Parameters

__new_size	Number of elements the list should contain.
__x	Data with which new elements should be populated.

This function will resize the list to the specified number of elements. If the number is smaller than the list's current size the list is truncated, otherwise the list is extended and new elements are populated with given data.
References std::end(), erase(), and insert().

reverse()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::reverse ()  [inline], [noexcept]
```
Reverse the elements in list.
Reverse the order of elements in the list in linear time.

size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::list< _Tp, _Alloc >::size () const  [inline], [nodiscard], [noexcept]
```
Returns the number of elements in the list.
Referenced by std::operator<=>(), remove(), remove_if(), unique(), and unique().

sort() [1/2]

```
template<typename _Tp, typename _Alloc>
```

```
void list::sort ()
```

Sort the elements.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

References std::begin(), std::empty(), and std::end().

sort() [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _StrictWeakOrdering>
void list::sort (
            _StrictWeakOrdering __comp)
```

Sort the elements according to comparison function.

Sorts the elements of this list in NlogN time. Equivalent elements remain in list order.

References std::begin(), std::empty(), and std::end().

splice() [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
            const_iterator __position,
            list< _Tp, _Alloc > && __x)  [inline], [noexcept]
```

Insert contents of another list.

Parameters

__position	Iterator referencing the element to insert before.
__x	Source list.

The elements of *__x* are inserted in constant time in front of the element referenced by *__position*. *__x* becomes an empty list.

Requires this != *__x*.

Referenced by insert(), insert(), remove(), remove_if(), unique(), and unique().

splice() [2/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
            const_iterator __position,
            list< _Tp, _Alloc > && __x,
            const_iterator __first,
            const_iterator __last)  [inline], [noexcept]
```

Insert range from another list.

Parameters

__position	Iterator referencing the element to insert before.
__x	Source list.
__first	Iterator referencing the start of range in x.
__last	Iterator referencing the end of range in x.

Removes elements in the range [__first,__last) and inserts them before __*position* in constant time. Undefined if __*position* is in [__first,__last).

splice() [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
            const_iterator __position,
            list< _Tp, _Alloc > && __x,
            const_iterator __i)  [inline], [noexcept]
```

Insert element from another list.

Parameters

__*position*	Iterator referencing the element to insert before.
__*x*	Source list.
__*i*	Iterator referencing the element to move.

Removes the element in list __*x* referenced by __*i* and inserts it into the current list before __*position*.

splice() [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
            const_iterator __position,
            list< _Tp, _Alloc > & __x,
            const_iterator __first,
            const_iterator __last)  [inline], [noexcept]
```

Insert range from another list.

Parameters

__*position*	Const_iterator referencing the element to insert before.
__*x*	Source list.
__*first*	Const_iterator referencing the start of range in x.
__*last*	Const_iterator referencing the end of range in x.

Removes elements in the range [__first,__last) and inserts them before __*position* in constant time. Undefined if __*position* is in [__first,__last).

splice() [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::splice (
            const_iterator __position,
```

```
            list< _Tp, _Alloc > & __x,
            const_iterator __i)  [inline], [noexcept]
```

Insert element from another list.

Parameters

__position	Iterator referencing the element to insert before.
__x	Source list.
__i	Iterator referencing the element to move.

Removes the element in list *__x* referenced by *__i* and inserts it into the current list before *__position*.

swap()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::list< _Tp, _Alloc >::swap (
            list< _Tp, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another list.

Parameters

_↩ _x	A list of the same element and allocator types.

This exchanges the elements between two lists in constant time. Note that the global std::swap() function is specialized such that std::swap(l1,l2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.

unique() [1/2]

```
template<typename _Tp, typename _Alloc>
list< _Tp, _Alloc >::__remove_return_type list::unique ()
```

Remove consecutive duplicate elements.
For each consecutive set of elements with the same value, remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References list(), std::begin(), begin(), std::end(), get_allocator(), size(), and splice().

unique() [2/2]

```
template<typename _Tp, typename _Alloc>
template<typename _BinaryPredicate>
list< _Tp, _Alloc >::__remove_return_type list::unique (
            _BinaryPredicate __binary_pred)
```

Remove consecutive elements satisfying a predicate.

Template Parameters

_BinaryPredicate	Binary predicate function or object.

For each consecutive set of elements [first,last) that satisfy predicate(first,i) where i is an iterator in [first,last), remove all but the first one. Remaining elements stay in list order. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.
References list(), std::begin(), begin(), std::end(), get_allocator(), size(), and splice().
The documentation for this class was generated from the following files:

- stl_list.h
- list.tcc

6.655 __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc > Class Template Reference

`#include <assoc_container.hpp>`

Public Types

- typedef list_update_tag **container_category**
- typedef Eq_Fn **eq_fn**
- typedef Update_Policy **update_policy**

Public Member Functions

- **list_update** (const list_update &other)
- template<typename It>
  list_update (It first, It last)
- list_update & **operator=** (const list_update &other)
- void **swap** (list_update &other)

6.655.1 Detailed Description

template<typename Key, typename Mapped, class Eq_Fn = typename detail::default_eq_fn<Key>::type, class Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>>
class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >

A list-update based associative container.

Template Parameters

Key	Key type.
Mapped	Map type.
Eq_Fn	Equal functor.
Update_Policy	Update policy, determines when an element will be moved to the front of the list.
_Alloc	Allocator type.

Base is detail::lu_map.

6.655.2 Constructor & Destructor Documentation

list_update()

```
template<typename Key, typename Mapped, class Eq_Fn = typename detail::default_eq_fn<Key>::type,
class Update_Policy = detail::default_update_policy::type, class _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >::list_update (
            It first,
            It last)  [inline]
```

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

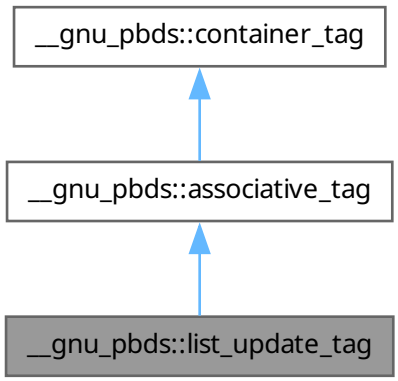
The documentation for this class was generated from the following file:

- assoc_container.hpp

6.656 __gnu_pbds::list_update_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::list_update_tag:



6.656.1 Detailed Description

List-update.

The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.657 std::locale Class Reference

```
#include <locale_classes.h>
```

Classes

- class facet
- class id

Public Types

- typedef int category

Public Member Functions

- locale () throw ()
- locale (const char *__s)
- locale (const locale &__base, const char *__s, category __cat)
- locale (const locale &__base, const locale &__add, category __cat)
- locale (const locale &__base, const std::string &__s, category __cat)
- locale (const locale &__other) throw ()
- template<typename _Facet>
  locale (const locale &__other, _Facet *__f)
- locale (const std::string &__s)
- ~locale () throw ()
- template<typename _Facet>
  locale combine (const locale &__other) const
- string name () const
- template<typename _Char, typename _Traits, typename _Alloc>
  bool operator() (const basic_string< _Char, _Traits, _Alloc > &__s1, const basic_string< _Char, _Traits, _Alloc > &__s2) const
- template<typename _CharT, typename _Traits, typename _Alloc>
  bool **operator()** (const basic_string< _CharT, _Traits, _Alloc > &__s1, const basic_string< _CharT, _Traits, _Alloc > &__s2) const
- const locale & operator= (const locale &__other) throw ()
- bool operator== (const locale &__other) const throw ()

Static Public Member Functions

- static const locale & classic ()
- static locale global (const locale &__loc)

Static Public Attributes

- static const category none
- static const category ctype
- static const category numeric
- static const category collate
- static const category time
- static const category monetary
- static const category messages
- static const category all

Friends

- template<typename _Facet>
  const _Facet * **__try_use_facet** (const locale &__loc)
- template<typename _Cache>
  struct **__use_cache**
- class **_Impl**
- class **facet**

- template<typename _Facet>
  bool has_facet (const locale &__loc) throw ()
- template<typename _Facet>
  const _Facet & use_facet (const locale &__loc)

6.657.1 Detailed Description

Container class for localization functionality.
The locale class is first a class wrapper for C library locales. It is also an extensible container for user-defined localization. A locale is a collection of facets that implement various localization features such as money, time, and number printing. Constructing C++ locales does not change the C library locale.
This library supports efficient construction and copying of locales through a reference counting implementation of the locale class.

6.657.2 Member Typedef Documentation

category

```
typedef int std::locale::category
```

Definition of locale::category.

6.657.3 Constructor & Destructor Documentation

locale() [1/8]

```
std::locale::locale () throw ( )
```

Default constructor.
Constructs a copy of the global locale. If no locale has been explicitly set, this is the C locale.
Referenced by locale(), locale(), locale(), locale(), locale(), locale(), locale(), ~locale(), combine(), global(), has_facet, operator()(), operator=(), operator==(), and use_facet.

locale() [2/8]

```
std::locale::locale (
            const locale & __other) throw ( )
```

Copy constructor.
Constructs a copy of *other*.

Parameters

__other	The locale to copy.

References locale().

locale() [3/8]

```
std::locale::locale (
            const char * __s)  [explicit]
```

Named locale constructor.
Constructs a copy of the named C library locale.

Parameters

_↩ _s	Name of the locale to construct.

Exceptions

std::runtime_error	if __s is null or an undefined locale.

References locale().

locale() [4/8]

```
std::locale::locale (
            const locale & __base,
            const char * __s,
            category __cat)
```

Construct locale with facets from another locale.
Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If base is named, this locale instance will also be named.

Parameters

__base	The locale to copy.
__s	Name of the locale to use facets from.
__cat	Set of categories defining the facets to use from __s.

Exceptions

std::runtime_error	if __s is null or an undefined locale.

References locale().

locale() [5/8]

```
std::locale::locale (
            const std::string & __s)  [inline], [explicit]
```

Named locale constructor.
Constructs a copy of the named C library locale.

Parameters

__s	Name of the locale to construct.

Exceptions

std::runtime_error	if __s is an undefined locale.

References locale().

locale() [6/8]

```
std::locale::locale (
```

```
            const locale & __base,
            const std::string & __s,
            category __cat)  [inline]
```

Construct locale with facets from another locale.
Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale named by *s*. If base is named, this locale instance will also be named.

Parameters

__base	The locale to copy.
__s	Name of the locale to use facets from.
__cat	Set of categories defining the facets to use from __s.

Exceptions

std::runtime_error	if __s is an undefined locale.

References locale().

locale() [7/8]

```
std::locale::locale (
            const locale & __base,
            const locale & __add,
            category __cat)
```

Construct locale with facets from another locale.
Constructs a copy of the locale *base*. The facets specified by *cat* are replaced with those from the locale *add*. If *base* and *add* are named, this locale instance will also be named.

Parameters

__base	The locale to copy.
__add	The locale to use facets from.
__cat	Set of categories defining the facets to use from add.

References locale().

locale() [8/8]

```
template<typename _Facet>
std::locale::locale (
            const locale & __other,
            _Facet * __f)
```

Construct locale with another facet.
Constructs a copy of the locale *__other*. The facet *__f* is added to *__other*, replacing an existing facet of type Facet if there is one. If *__f* is null, this locale is a copy of *__other*.

Parameters

__other	The locale to copy.
__f	The facet to add in.

References locale().

~locale()

```
std::locale::~locale () throw ( )
```
Locale destructor.
References locale().

6.657.4 Member Function Documentation

classic()

```
const locale & std::locale::classic ()  [static], [nodiscard]
```
Return reference to the C locale.

combine()

```
template<typename _Facet>
locale std::locale::combine (
            const locale & __other) const  [nodiscard]
```
Construct locale with another facet.
Constructs and returns a new copy of this locale. Adds or replaces an existing facet of type Facet from the locale *other* into the new locale.

Template Parameters

_Facet	The facet type to copy from other

Parameters

__other	The locale to copy from.

Returns

Newly constructed locale.

Exceptions

std::runtime_error	if __other has no facet of type _Facet.

References locale().
Referenced by operator=().

global()

```
locale std::locale::global (
            const locale & __loc)  [static]
```

Set global locale.

This function sets the global locale to the argument and returns a copy of the previous global locale. If the argument has a name, it will also call std::setlocale(LC_ALL, loc.name()).

Parameters

__loc	The new locale to make global.

Returns

Copy of the old global locale.

References locale().

name()

```
string std::locale::name () const  [nodiscard]
```

Return locale name.

Returns

Locale name or "*" if unnamed.

References name().
Referenced by name().

operator()()

```
template<typename _Char, typename _Traits, typename _Alloc>
bool std::locale::operator() (
            const basic_string< _Char, _Traits, _Alloc > & __s1,
            const basic_string< _Char, _Traits, _Alloc > & __s2) const  [nodiscard]
```

Compare two strings according to collate.

Template operator to compare two strings using the compare function of the collate facet in this locale. One use is to provide the locale to the sort function. For example, a vector v of strings could be sorted according to locale loc by doing:

```
std::sort(v.begin(), v.end(), loc);
```

Parameters

__s1	First string to compare.
__s2	Second string to compare.

Returns

True if collate<_Char> facet compares __s1 < __s2, else false.

References locale().

operator=()

```
const locale & std::locale::operator= (
            const locale & __other) throw ( )
```

Assignment operator.
Set this locale to be a copy of *other*.

Parameters

__other	The locale to copy.

Returns

A reference to this locale.

References locale(), and combine().

operator==()

```
bool std::locale::operator== (
            const locale & __other) const throw ( )   [nodiscard]
```

Locale equality.

Parameters

__other	The locale to compare against.

Returns

True if other and this refer to the same locale instance, are copies, or have the same name. False otherwise.

References locale(), and operator==().
Referenced by operator==().

6.657.5 Friends And Related Symbol Documentation

has_facet

```
template<typename _Facet>
bool has_facet (
            const locale & __loc) throw ( )   [friend]
```

Test for the presence of a facet.
has_facet tests the locale argument for the presence of the facet type provided as the template parameter. Facets derived from the facet parameter will also return true.

Template Parameters

_Facet	The facet type to test the presence of.

Parameters

__loc	The locale to test.

Returns

true if __loc contains a facet of type _Facet, else false.

References locale().

use_facet

```
template<typename _Facet>
const _Facet & use_facet (
             const locale & __loc)  [friend]
```

Return a facet.
use_facet looks for and returns a reference to a facet of type Facet where Facet is the template parameter. If has_facet(locale) is true, there is a suitable facet to return. It throws std::bad_cast if the locale doesn't contain a facet of type Facet.

Template Parameters

_Facet	The facet type to access.

Parameters

__loc	The locale to use.

Returns

Reference to facet of type Facet.

Exceptions

std::bad_cast	if __loc doesn't contain a facet of type _Facet.

References locale().

6.657.6 Member Data Documentation

all

```
const category std::locale::all  [static]
```

Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc

collate

```
const category std::locale::collate  [static]
```

Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc

ctype

```
const category std::locale::ctype  [static]
```
Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc

messages

```
const category std::locale::messages  [static]
```
Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc

monetary

```
const category std::locale::monetary  [static]
```
Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc

none

```
const category std::locale::none  [static]
```
Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc

numeric

```
const category std::locale::numeric  [static]
```
Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc

time

```
const category std::locale::time  [static]
```
Category values.
The standard category values are none, ctype, numeric, collate, time, monetary, and messages. They form a bitmask that supports union and intersection. The category all is the union of these values.
NB: Order must match _S_facet_categories definition in locale.cc
The documentation for this class was generated from the following files:

- locale_classes.h
- locale_classes.tcc

6.658 std::lock_guard< _Mutex > Class Template Reference

```
#include <mutex>
```

Public Types

- typedef _Mutex **mutex_type**

Public Member Functions

- **lock_guard** (const lock_guard &)=delete
- **lock_guard** (mutex_type &__m)
- **lock_guard** (mutex_type &__m, adopt_lock_t) noexcept
- lock_guard & **operator=** (const lock_guard &)=delete

6.658.1 Detailed Description

template<**typename _Mutex**>
class std::lock_guard< **_Mutex** >

A simple scoped lock type.
A lock_guard controls mutex ownership within a scope, releasing ownership in the destructor.

Since

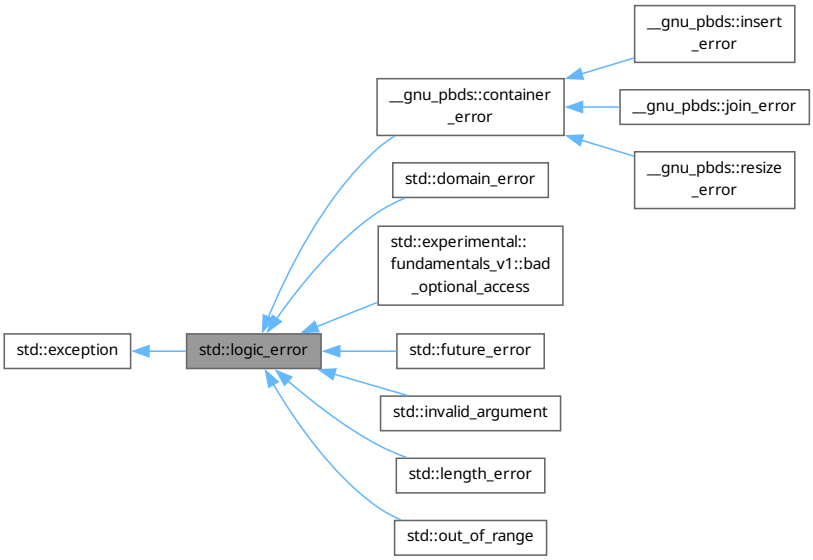
C++11

The documentation for this class was generated from the following file:

- std_mutex.h

6.659 std::logic_error Class Reference

`#include <stdexcept_except.h>`
Inheritance diagram for std::logic_error:



Public Member Functions

- **logic_error** (const char *)
- **logic_error** (const logic_error &) noexcept
- logic_error (const string &__arg)
- **logic_error** (logic_error &&) noexcept
- logic_error & **operator=** (const logic_error &) noexcept
- logic_error & **operator=** (logic_error &&) noexcept
- virtual const char * what () const noexcept

6.659.1 Detailed Description

One of two subclasses of exception.
Logic errors represent problems in the internal logic of a program; in theory, these are preventable, and even detectable before the program runs (e.g., violations of class invariants).

6.659.2 Constructor & Destructor Documentation

logic_error()

```
std::logic_error::logic_error (
            const string & __arg)  [explicit]
```

Takes a character string describing the error.
References logic_error().
Referenced by logic_error().

6.659.3 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept]
```

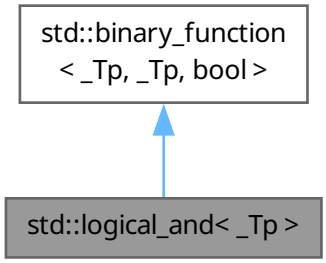
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
References what().
Referenced by what().
The documentation for this class was generated from the following file:

- stdexcept_except.h

6.660 std::logical_and< _Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::logical_and< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

6.660.1 Detailed Description

template<typename _Tp>
struct std::logical_and< _Tp >

One of the Boolean operations functors.

6.660.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

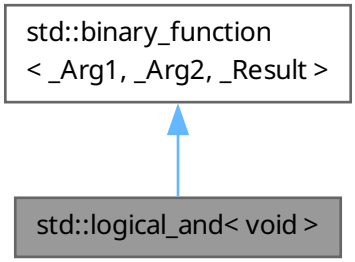
second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.661 std::logical_and< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::logical_and< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) &&std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) &&std::forward< _Up >(__u))

6.661.1 Detailed Description

One of the Boolean operations functors.

6.661.2 Member Typedef Documentation

first_argument_type

`typedef void std::binary_function< void, void, bool >::first_argument_type`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< void, void, bool >::result_type`
`result_type` is the return type

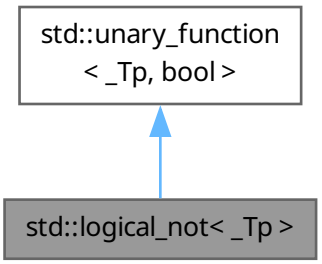
second_argument_type

`typedef void std::binary_function< void, void, bool >::second_argument_type`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.662 std::logical_not< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::logical_not< _Tp >:



Public Types

- typedef _Tp argument_type
- typedef bool result_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x) const

6.662.1 Detailed Description

template<typename _Tp>
struct std::logical_not< _Tp >

One of the Boolean operations functors.

6.662.2 Member Typedef Documentation

argument_type

`typedef _Tp std::unary_function< _Tp, bool >::argument_type [inherited]`
`argument_type` is the type of the argument

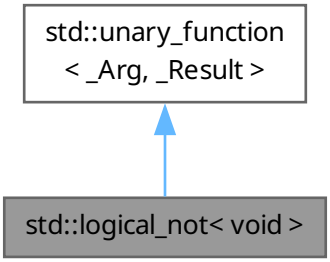
result_type

`typedef bool std::unary_function< _Tp, bool >::result_type [inherited]`
`result_type` is the return type
The documentation for this struct was generated from the following file:

- stl_function.h

6.663 std::logical_not< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::logical_not< void >:



Public Types

- typedef void argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type

Public Member Functions

- template<typename _Tp>
  constexpr auto **operator()** (_Tp &&__t) const noexcept(noexcept(!std::forward< _Tp >(__t))) -> decltype(!std::forward< _Tp >(__t))

6.663.1 Detailed Description

One of the Boolean operations functors.

6.663.2 Member Typedef Documentation

argument_type

`typedef void std::unary_function< void, bool >::argument_type`
`argument_type` is the type of the argument

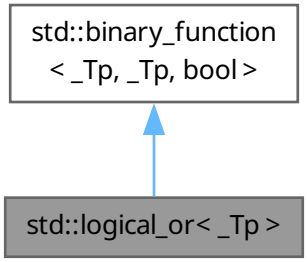
result_type

`typedef bool std::unary_function< void, bool >::result_type`
`result_type` is the return type
The documentation for this struct was generated from the following file:

- stl_function.h

6.664 std::logical_or< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::logical_or< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

6.664.1 Detailed Description

template<typename _Tp>
struct std::logical_or< _Tp >

One of the Boolean operations functors.

6.664.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

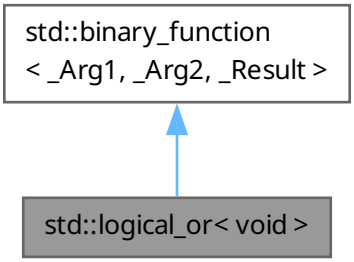
second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.665 std::logical_or< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::logical_or< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t)||std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t)||std::forward< _Up >(__u))

6.665.1 Detailed Description

One of the Boolean operations functors.

6.665.2 Member Typedef Documentation

first_argument_type

`typedef void std::binary_function< void, void, bool >::first_argument_type`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< void, void, bool >::result_type`
`result_type` is the return type

second_argument_type

`typedef void std::binary_function< void, void, bool >::second_argument_type`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.666 std::lognormal_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **lognormal_distribution** (_RealType __m, _RealType __s=_RealType(1))
- **lognormal_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType **m** () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()
- _RealType **s** () const

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::lognormal_distribution< _RealType1 > &__x)
- bool operator== (const lognormal_distribution &__d1, const lognormal_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::lognormal_distribution< _RealType1 > &__x)

6.666.1 Detailed Description

template<typename _RealType = double>
class std::lognormal_distribution< _RealType >

A lognormal_distribution random number distribution.
The formula for the normal probability mass function is

$$p(x|m,s) = \frac{1}{sx\sqrt{2\pi}} \exp -\frac{(\ln x - m)^2}{2s^2}$$

Since

C++11

6.666.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::lognormal_distribution< _RealType >::result_type
```

The type of the range of the distribution.

6.666.3 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::lognormal_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::lognormal_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::lognormal_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::lognormal_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::lognormal_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::lognormal_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

6.666.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::lognormal_distribution< _RealType1 > & __x)  [friend]
```

Inserts a lognormal_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A lognormal_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
            const lognormal_distribution< _RealType > & __d1,
            const lognormal_distribution< _RealType > & __d2)  [friend]
```

Return true if two lognormal distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::lognormal_distribution< _RealType1 > & __x)  [friend]
```

Extracts a lognormal_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A lognormal_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.667 __gnu_pbds::detail::lu_counter_metadata< Size_Type > Class Template Reference

`#include <lu_counter_metadata.hpp>`

Public Types

- typedef Size_Type **size_type**

Friends

- class **lu_counter_policy_base**< **size_type** >

6.667.1 Detailed Description

template<**typename Size_Type = std::size_t**>
class __gnu_pbds::detail::lu_counter_metadata< **Size_Type** >

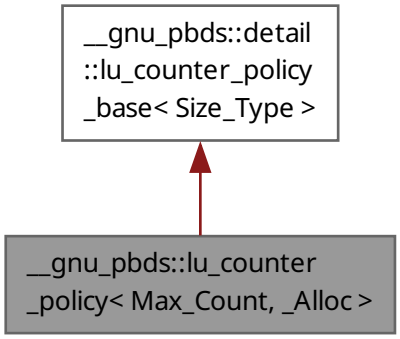
A list-update metadata type that moves elements to the front of the list based on the counter algorithm.
The documentation for this class was generated from the following file:

- lu_counter_metadata.hpp

6.668 __gnu_pbds::lu_counter_policy< Max_Count, _Alloc > Class Template Reference

`#include <list_update_policy.hpp>`

Inheritance diagram for __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >:



Public Types

- enum { max_count }
- typedef _Alloc **allocator_type**
- typedef detail::rebind_traits< _Alloc, metadata_type >::reference metadata_reference
- typedef detail::lu_counter_metadata< size_type > metadata_type
- typedef allocator_type::size_type **size_type**

Public Member Functions

- metadata_type operator() () const
- bool operator() (metadata_reference r_data) const

Private Member Functions

- template<typename Metadata_Reference>
  bool **operator()** (Metadata_Reference r_data, size_type m_max_count) const
- lu_counter_metadata< size_type > **operator()** (size_type max_size) const

6.668.1 Detailed Description

template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
class __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >

A list-update policy that moves elements to the front of the list based on the counter algorithm.

6.668.2 Member Typedef Documentation

metadata_reference

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
typedef detail::rebind_traits<_Alloc,metadata_type>::reference __gnu_pbds::lu_counter_policy<
Max_Count, _Alloc >::metadata_reference
```

Reference to metadata on which this functor operates.

metadata_type

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
typedef detail::lu_counter_metadata<size_type> __gnu_pbds::lu_counter_policy< Max_Count, _Alloc
>::metadata_type
```

Metadata on which this functor operates.

6.668.3 Member Enumeration Documentation

anonymous enum

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
anonymous enum
```

Enumerator

max_count	When some element is accessed this number of times, it will be moved to the front of the list.

6.668.4 Member Function Documentation

operator()() [1/2]

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
metadata_type __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() () const  [inline]
```

Creates a metadata object.

References max_count.

operator()() [2/2]

```
template<std::size_t Max_Count = 5, typename _Alloc = std::allocator<char>>
bool __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >::operator() (
            metadata_reference r_data) const  [inline]
```

Decides whether a metadata object should be moved to the front of the list.

References max_count.

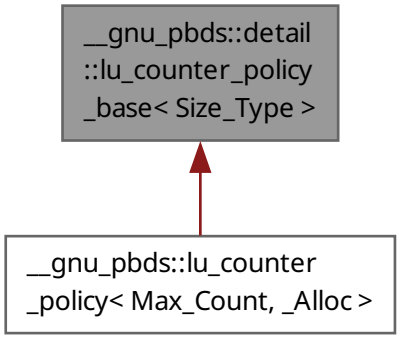
The documentation for this class was generated from the following file:

- list_update_policy.hpp

6.669 __gnu_pbds::detail::lu_counter_policy_base< Size_Type > Class Template Reference

```
#include <lu_counter_metadata.hpp>
```

Inheritance diagram for __gnu_pbds::detail::lu_counter_policy_base< Size_Type >:



Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- template<typename Metadata_Reference>
  bool **operator()** (Metadata_Reference r_data, size_type m_max_count) const
- lu_counter_metadata< size_type > **operator()** (size_type max_size) const

6.669.1 Detailed Description

template<typename Size_Type>
class __gnu_pbds::detail::lu_counter_policy_base< Size_Type >

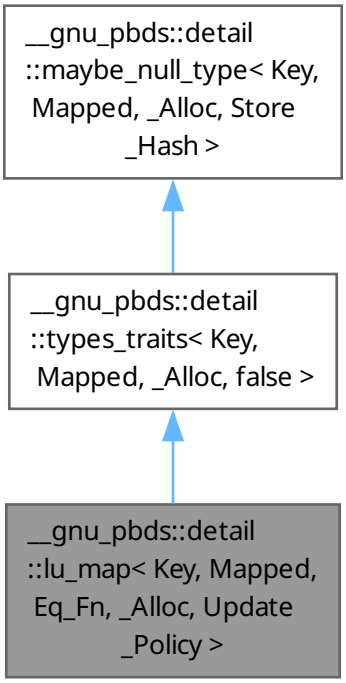
Base class for list-update counter policy.
The documentation for this class was generated from the following file:

- lu_counter_metadata.hpp

6.670 __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > Class Template Reference

`#include <lu_map_.hpp>`

Inheritance diagram for __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >:



Public Types

- typedef _Alloc **allocator_type**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef const_iterator_ **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef Eq_Fn **eq_fn**
- typedef iterator_ **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**

- typedef point_const_iterator_ **point_const_iterator**
- typedef point_iterator_ **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef Update_Policy::metadata_type **update_metadata**
- typedef Update_Policy **update_policy**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **lu_map** (const lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > &)
- template<typename It>
  lu_map (It, It)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- bool **erase** (key_const_reference)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference r_key)
- point_const_iterator **find** (key_const_reference r_key) const
- std::pair< point_iterator, bool > **insert** (const_reference)
- size_type **max_size** () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- size_type **size** () const
- void **swap** (lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy > &)

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Protected Member Functions

- template<typename It>
  void **copy_from_range** (It, It)

Friends

- class **const_iterator_**
- class **iterator_**

6.670.1 Detailed Description

template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
class __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >

list-based (with updates) associative container. Skip to the lu, my darling.
The documentation for this class was generated from the following file:

- lu_map_.hpp

6.671 __gnu_pbds::lu_move_to_front_policy< _Alloc > Class Template Reference

`#include <list_update_policy.hpp>`

Public Types

- typedef _Alloc **allocator_type**
- typedef detail::rebind_traits< _Alloc, metadata_type >::reference metadata_reference
- typedef null_type metadata_type

Public Member Functions

- metadata_type operator() () const
- bool operator() (metadata_reference r_metadata) const

6.671.1 Detailed Description

template<typename _Alloc = std::allocator<char>>
class __gnu_pbds::lu_move_to_front_policy< _Alloc >

A list-update policy that unconditionally moves elements to the front of the list. A null type means that each link in a list-based container does not actually need metadata.

6.671.2 Member Typedef Documentation

metadata_reference

```
template<typename _Alloc = std::allocator<char>>
typedef detail::rebind_traits<_Alloc,metadata_type>::reference __gnu_pbds::lu_move_to_front_policy<
_Alloc >::metadata_reference
```

Reference to metadata on which this functor operates.

metadata_type

```
template<typename _Alloc = std::allocator<char>>
typedef null_type __gnu_pbds::lu_move_to_front_policy< _Alloc >::metadata_type
```

Metadata on which this functor operates.

6.671.3 Member Function Documentation

operator()() [1/2]

```
template<typename _Alloc = std::allocator<char>>
metadata_type __gnu_pbds::lu_move_to_front_policy< _Alloc >::operator() () const  [inline]
```

Creates a metadata object.

operator()() [2/2]

```
template<typename _Alloc = std::allocator<char>>
bool __gnu_pbds::lu_move_to_front_policy< _Alloc >::operator() (
            metadata_reference r_metadata) const  [inline]
```

Decides whether a metadata object should be moved to the front of the list.
The documentation for this class was generated from the following file:

- list_update_policy.hpp

6.672 std::make_signed< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.672.1 Detailed Description

template<typename _Tp>
struct std::make_signed< _Tp >

make_signed
The documentation for this struct was generated from the following file:

- type_traits

6.673 std::make_unsigned< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.673.1 Detailed Description

template<typename _Tp>
struct std::make_unsigned< _Tp >

make_unsigned
The documentation for this struct was generated from the following file:

- type_traits

6.674 __gnu_cxx::malloc_allocator< _Tp > Class Template Reference

`#include <malloc_allocator.h>`

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr **malloc_allocator** (const malloc_allocator &) noexcept
- template<typename _Tp1>
  constexpr **malloc_allocator** (const malloc_allocator< _Tp1 > &) noexcept
- _Tp * **allocate** (size_type __n, const void *=0)
- void **deallocate** (_Tp *__p, size_type)

Friends

- template<typename _Up>
  constexpr bool **operator==** (const malloc_allocator &, const malloc_allocator< _Up > &) noexcept

6.674.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::malloc_allocator< _Tp >

An allocator that uses malloc.
This is precisely the allocator defined in the C++ Standard.

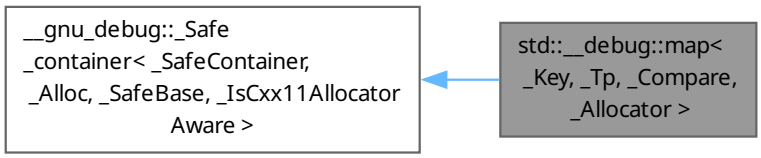
- all allocation calls malloc
- all deallocation calls free

The documentation for this class was generated from the following files:

- experimental/memory_resource
- malloc_allocator.h

6.675 std::__debug::map< _Key, _Tp, _Compare, _Allocator > Class Template Reference

`#include <map.h>`
Inheritance diagram for std::__debug::map< _Key, _Tp, _Compare, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, map > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, map > **iterator**

- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef std::pair< const _Key, _Tp > **value_type**

Public Member Functions

- **map** (_Base_ref __x)
- template<typename _InputIterator>
  map (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &_↩
  _a=_Allocator())
- template<typename _InputIterator>
  map (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- **map** (const _Compare &__comp, const _Allocator &__a=_Allocator())
- **map** (const allocator_type &__a)
- **map** (const map &)=default
- **map** (const map &__m, const __type_identity_t< allocator_type > &__a)
- **map** (initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__↩
  a=allocator_type())
- **map** (initializer_list< value_type > __l, const allocator_type &__a)
- **map** (map &&)=default
- **map** (map &&__m, const __type_identity_t< allocator_type > &__a) noexcept(noexcept(_Base(std::move(__m), __a)))
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- _Base_iterator **erase** (_Base_const_iterator __position)
- template<__heterogeneous_tree_key< map > _Kt>
  size_type **erase** (_Kt &&__x)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __position)
- iterator **erase** (iterator __position)
- iterator **find** (const key_type &__x)

- const_iterator **find** (const key_type &__x) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  std::pair< iterator, bool > **insert** (_Pair &&__x)
- std::pair< iterator, bool > **insert** (const value_type &__x)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __position, _Pair &&__x)
- iterator **insert** (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- void **insert** (std::initializer_list< value_type > __list)
- std::pair< iterator, bool > **insert** (value_type &&__x)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- map & **operator=** (const map &)=default
- map & **operator=** (initializer_list< value_type > __l)
- map & **operator=** (map &&)=default
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rend** () const noexcept
- reverse_iterator **rend** () noexcept
- void **swap** (map &__x) noexcept(/*conditional */)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

6.675.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp> > >
class std::__debug::map< _Key, _Tp, _Compare, _Allocator >

Class std::map with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- map.h

6.676 std::map< _Key, _Tp, _Compare, _Alloc > Class Template Reference

`#include <map>`

Public Types

- typedef _Alloc **allocator_type**
- typedef _Rep_type::const_iterator **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef _Rep_type::const_reverse_iterator **const_reverse_iterator**
- typedef _Rep_type::difference_type **difference_type**
- typedef _Rep_type::iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- typedef _Alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef _Rep_type::reverse_iterator **reverse_iterator**
- typedef _Rep_type::size_type **size_type**
- typedef std::pair< const _Key, _Tp > **value_type**

Public Member Functions

- map ()=default
- template<typename _InputIterator>
  map (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  map (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__↩
  a=allocator_type())
- template<typename _InputIterator>
  map (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- map (const _Compare &__comp, const allocator_type &__a=allocator_type())
- map (const allocator_type &__a)
- map (const map &)=default
- map (const map &__m, const __type_identity_t< allocator_type > &__a)
- map (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__↩
  a=allocator_type())
- map (initializer_list< value_type > __l, const allocator_type &__a)
- map (map &&)=default
- map (map &&__m, const __type_identity_t< allocator_type > &__a) noexcept(is_nothrow_copy_constructible< _Compare >::value &&_Alloc_traits::_S_always_equal())
- ~map ()=default
- template<__heterogeneous_tree_key< map > _Kt>
  mapped_type & **at** (const _Kt &__k)
- template<__heterogeneous_tree_key< map > _Kt>
  const mapped_type & **at** (const _Kt &__k) const
- mapped_type & at (const key_type &__k)
- const mapped_type & **at** (const key_type &__k) const
- const_iterator begin () const noexcept
- iterator begin () noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept

- template<typename... _Args>
  std::pair< iterator, bool > emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- const_iterator end () const noexcept
- iterator end () noexcept
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (std::initializer_list< value_type > __list)
- key_compare key_comp () const
- size_type max_size () const noexcept
- map & operator= (const map &)=default
- map & operator= (initializer_list< value_type > __l)
- map & operator= (map &&)=default
- template<__heterogeneous_tree_key< map > _Kt>
  mapped_type & **operator[ ]** (_Kt &&__k)
- mapped_type & operator[ ] (const key_type &__k)
- mapped_type & **operator[ ]** (key_type &&__k)
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rend () const noexcept
- reverse_iterator rend () noexcept
- size_type size () const noexcept
- void swap (map &__x) noexcept(/*conditional */)
- value_compare value_comp () const

- std::pair< iterator, bool > insert (const value_type &__x)
- std::pair< iterator, bool > insert (value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair >::value, pair< iterator, bool > > insert (_Pair &&__x)

- iterator insert (const_iterator __position, const value_type &__x)
- iterator insert (const_iterator __position, value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair >::value, iterator > insert (const_iterator __position, _Pair &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __position)

- size_type erase (const key_type &__x)
- template<__heterogeneous_tree_key< map > _Kt>
  size_type erase (_Kt &&__x)

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const
- size_type count (const key_type &__x) const
- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_t._M_find_tr(__x), void(), true)
- iterator lower_bound (const key_type &__x)
- const_iterator lower_bound (const key_type &__x) const
- iterator upper_bound (const key_type &__x)
- const_iterator upper_bound (const key_type &__x) const
- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const

Friends

- template<typename _K1, typename _T1, typename _C1, typename _A1>
  __detail::__synth3way_t< pair< const _K1, _T1 > > **operator<=>** (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)
- template<typename _K1, typename _T1, typename _C1, typename _A1>
  bool **operator==** (const map< _K1, _T1, _C1, _A1 > &, const map< _K1, _T1, _C1, _A1 > &)

6.676.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >>
class std::map< _Key, _Tp, _Compare, _Alloc >

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Since

C++98

Template Parameters

_Key	Type of key objects.
_Tp	Type of mapped objects.
_Compare	Comparison function object type, defaults to less<_Key>.
_Alloc	Allocator type, defaults to allocator<pair<const _Key, _Tp>.

Meets the requirements of a container, a reversible container, and an associative container (using unique keys). For a map<Key,T> the key_type is Key, the mapped_type is T, and the value_type is std::pair<const Key,T>.
Maps support bidirectional iterators.
The private tree data is declared exactly the same way for map and multimap; the distinction is made entirely in how the tree functions are called (*_unique versus *_equal, same as the standard).

6.676.2 Constructor & Destructor Documentation

map() [1/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map ()  [default]
```

Default constructor creates no elements.

map() [2/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Creates a map with no elements.

Parameters

__comp	A comparison object.
__a	An allocator object.

map() [3/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            const map< _Key, _Tp, _Compare, _Alloc > & )  [default]
```

Map copy constructor.
Whether the allocator is copied depends on the allocator traits.

map() [4/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            map< _Key, _Tp, _Compare, _Alloc > && )  [default]
```

Map move constructor.
The newly-created map contains the exact contents of the moved instance. The moved instance is a valid, but unspecified, map.

map() [5/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            initializer_list< value_type > __l,
            const _Compare & __comp = _Compare(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a map from an initializer_list.

Parameters

__l	An initializer_list.
__comp	A comparison object.
__a	An allocator object.

Create a map consisting of copies of the elements in the initializer_list *__l*. This is linear in N if the range is already sorted, and NlogN otherwise (where N is *__l.size()*).

map() [6/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            const allocator_type & __a)  [inline], [explicit]
```

Allocator-extended default constructor.

map() [7/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            const map< _Key, _Tp, _Compare, _Alloc > & __m,
            const __type_identity_t< allocator_type > & __a)  [inline]
```

Allocator-extended copy constructor.

map() [8/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
```

```
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            map< _Key, _Tp, _Compare, _Alloc > && __m,
            const __type_identity_t< allocator_type > & __a)  [inline], [noexcept]
```

Allocator-extended move constructor.

map() [9/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            initializer_list< value_type > __l,
            const allocator_type & __a)  [inline]
```

Allocator-extended initializer-list constructor.

map() [10/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a)  [inline]
```

Allocator-extended range constructor.

map() [11/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Builds a map from a range.

Parameters

__first	An input iterator.
__last	An input iterator.

Create a map consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

map() [12/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::map< _Key, _Tp, _Compare, _Alloc >::map (
            _InputIterator __first,
            _InputIterator __last,
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a map from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__comp	A comparison functor.
__a	An allocator object.

Create a map consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

~map()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::map< _Key, _Tp, _Compare, _Alloc >::~map ()  [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

6.676.3 Member Function Documentation

at()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
mapped_type & std::map< _Key, _Tp, _Compare, _Alloc >::at (
            const key_type & __k)  [inline]
```

Access to map data.

Parameters

_↩ _k	The key for which data should be retrieved.

Returns

A reference to the data whose key is equivalent to *__k*, if such a data is present in the map.

Exceptions

std::out_of_range	If no such data is present.

If a heterogeneous key __k matches a range of elements, the first is chosen.

begin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::begin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

begin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::begin ()  [inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

cbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the map. Iteration is done in ascending order according to the keys.

cend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

clear()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map< _Key, _Tp, _Compare, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in a map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::map< _Key, _Tp, _Compare, _Alloc >::contains (
             const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void(), true)  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

True if there is an element with the specified key.

contains() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::map< _Key, _Tp, _Compare, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

True if there is an element with the specified key.

count()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements with given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Number of elements with specified key.

This function only makes sense for multimaps; for map the result will either be 0 (not present) or 1 (present).

crbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::crbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

crend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::crend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

emplace()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
std::pair< iterator, bool > std::map< _Key, _Tp, _Compare, _Alloc >::emplace (
            _Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the map.

Parameters

__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.
Insertion requires logarithmic time.

emplace_hint()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&... __args) [inline]
```

Attempts to build and insert a std::pair into the map.

Parameters

__pos	An iterator that serves as a hint as to where the pair should be inserted.
__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the element with key of the std::pair built from *__args* (may or may not be that std::pair).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires logarithmic time (if the hint is not taken).
Referenced by std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::emplace(), and std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert().

empty()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::map< _Key, _Tp, _Compare, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the map is empty. (Thus begin() would equal end().)

end() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

end() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::end ()  [inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the map. Iteration is done in ascending order according to the keys.

Referenced by std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::at(), std::map< _Key, _Tp, _Cmp, po
std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), and std::map< _Key, _Tp, _Cmp, polymorp

equal_range() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< iterator, iterator > std::map< _Key, _Tp, _Compare, _Alloc >::equal_range (
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

This function probably only makes sense for multimaps.

equal_range() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< const_iterator, const_iterator > std::map< _Key, _Tp, _Compare, _Alloc >::equal_range
(
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multimaps.

erase() [1/5]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<__heterogeneous_tree_key< map > _Kt>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::erase (
             _Kt && __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/5]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::erase (
             const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/5]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [first,last) range of elements from a map.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [4/5]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from a map.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [5/5]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
iterator std::map< _Key, _Tp, _Compare, _Alloc >::erase (
            iterator __position)  [inline]
```

Erases an element from a map.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in a map.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in a map.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
allocator_type std::map< _Key, _Tp, _Compare, _Alloc >::get_allocator () const  [inline], [noexcept]
```

Get a copy of the memory allocation object.

insert() [1/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
void std::map< _Key, _Tp, _Compare, _Alloc >::insert (
             _InputIterator __first,
             _InputIterator __last)  [inline]
```

Template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair >::value, pair< iterator, bool > > std::map<
_Key, _Tp, _Compare, _Alloc >::insert (
             _Pair && __x)  [inline]
```

Attempts to insert a std::pair into the map.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.
Insertion requires logarithmic time.

insert() [3/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< iterator, bool > std::map< _Key, _Tp, _Compare, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Attempts to insert a std::pair into the map.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.
Insertion requires logarithmic time.
Referenced by std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), and std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::operator[]().

insert() [4/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair >::value, iterator > std::map< _Key, _Tp, _↩
Compare, _Alloc >::insert (
            const_iterator __position,
            _Pair && __x)  [inline]
```

Attempts to insert a std::pair into the map.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↩ associative.insert_hints for more on *hinting*.
Insertion requires logarithmic time (if the hint is not taken).

insert() [5/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::insert (
            const_iterator __position,
            const value_type & __x)  [inline]
```

Attempts to insert a std::pair into the map.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires logarithmic time (if the hint is not taken).

insert() [6/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::insert (
            const_iterator __position,
            value_type && __x)  [inline]
```

Attempts to insert a std::pair into the map.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires logarithmic time (if the hint is not taken).

insert() [7/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map< _Key, _Tp, _Compare, _Alloc >::insert (
            std::initializer_list< value_type > __list)  [inline]
```

Attempts to insert a list of std::pairs into the map.

Parameters

__list	A std::initializer_list<value_type> of pairs to be inserted.

Complexity similar to that of the range constructor.

insert() [8/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< iterator, bool > std::map< _Key, _Tp, _Compare, _Alloc >::insert (
            value_type && __x)  [inline]
```

Attempts to insert a std::pair into the map.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the map. A map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the map.
Insertion requires logarithmic time.

key_comp()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
key_compare std::map< _Key, _Tp, _Compare, _Alloc >::key_comp () const  [inline]
```

Returns the key comparison object out of which the map was constructed.
Referenced by std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::at(), std::map< _Key, _Tp, _Cmp, po
std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), and std::map< _Key, _Tp, _Cmp, polymorp

lower_bound() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
            const key_type & __x)  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.
Referenced by std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::at(), std::map< _Key, _Tp, _Cmp, po
std::map< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert(), and std::map< _Key, _Tp, _Cmp, polymorp

lower_bound() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::lower_bound (
            const key_type & __x) const  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

max_size()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the maximum size of the map.

operator=() [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map & std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
            const map< _Key, _Tp, _Compare, _Alloc > & )  [default]
```

Map assignment operator.
Whether the allocator is copied depends on the allocator traits.

operator=() [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
map & std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Map list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a map with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the map and that the resulting map's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
map & std::map< _Key, _Tp, _Compare, _Alloc >::operator= (
            map< _Key, _Tp, _Compare, _Alloc > && )  [default]
```

Move assignment operator.

operator[]()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
mapped_type & std::map< _Key, _Tp, _Compare, _Alloc >::operator[] (
            const key_type & __k)  [inline]
```

Subscript ([]) access to map data.

Parameters

_↩ *_k*	The key for which data should be retrieved.

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([]) operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.
If a heterogeneous key matches a range of elements, the first is chosen.
Lookup requires logarithmic time.

rbegin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

rbegin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rbegin ()  [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the map. Iteration is done in descending order according to the keys.

rend() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

rend() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::map< _Key, _Tp, _Compare, _Alloc >::rend ()  [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the map. Iteration is done in descending order according to the keys.

size()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::map< _Key, _Tp, _Compare, _Alloc >::size () const  [inline], [noexcept]
```

Returns the size of the map.

swap()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::map< _Key, _Tp, _Compare, _Alloc >::swap (
            map< _Key, _Tp, _Compare, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another map.

Parameters

_↩ _x	A map of the same element and allocator types.

This exchanges the elements between two maps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.

upper_bound() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
            const key_type & __x)  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::map< _Key, _Tp, _Compare, _Alloc >::upper_bound (
            const key_type & __x) const  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

value_comp()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
value_compare std::map< _Key, _Tp, _Compare, _Alloc >::value_comp () const  [inline]
```

Returns a value comparison object, built from the key comparison object out of which the map was constructed.
The documentation for this class was generated from the following file:

- stl_map.h

6.677 std::mask_array< _Tp > Class Template Reference

```
#include <mask_array.h>
```

Public Types

- typedef _Tp **value_type**

Public Member Functions

- mask_array (const mask_array &)
- template<class _Dom>
  void **operator%=** (const _Expr< _Dom, _Tp > &) const
- void operator%= (const valarray< _Tp > &) const

- template<class _Dom>
  void **operator&=** (const _Expr< _Dom, _Tp > &) const
- void operator&= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator*=** (const _Expr< _Dom, _Tp > &) const
- void operator*= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator+=** (const _Expr< _Dom, _Tp > &) const
- void operator+= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator-=** (const _Expr< _Dom, _Tp > &) const
- void operator-= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator/=** (const _Expr< _Dom, _Tp > &) const
- void operator/= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator<<=** (const _Expr< _Dom, _Tp > &) const
- void operator<<= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator=** (const _Expr< _Dom, _Tp > &) const
- template<class _Ex>
  void **operator=** (const _Expr< _Ex, _Tp > &__e) const
- void operator= (const _Tp &) const
- mask_array & operator= (const mask_array &)
- void **operator=** (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator>>=** (const _Expr< _Dom, _Tp > &) const
- void operator>>= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator^=** (const _Expr< _Dom, _Tp > &) const
- void operator^= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator|=** (const _Expr< _Dom, _Tp > &) const
- void operator|= (const valarray< _Tp > &) const

Friends

- class **valarray**< **_Tp** >

6.677.1 Detailed Description

template<**class _Tp**>
class std::mask_array< **_Tp** >

Reference to selected subset of an array.
A mask_array is a reference to the actual elements of an array specified by a bitmask in the form of an array of bool. The way to get a mask_array is to call operator[](valarray<bool>) on a valarray. The returned mask_array then permits carrying operations out on the referenced subset of elements in the original valarray.
For example, if a mask_array is obtained using the array (false, true, false, true) as an argument, the mask array has two elements referring to array[1] and array[3] in the underlying array.

Parameters

Tp	Element type.

6.677.2 Member Function Documentation

operator%=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator%= (
            const valarray< _Tp > & ) const
```

Modulo slice elements by corresponding elements of *v*.

operator&=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator&= (
            const valarray< _Tp > & ) const
```

Logical and slice elements with corresponding elements of *v*.

operator*=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator*= (
            const valarray< _Tp > & ) const
```

Multiply slice elements by corresponding elements of *v*.

operator+=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator+= (
            const valarray< _Tp > & ) const
```

Add corresponding elements of *v* to slice elements.

operator-=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator-= (
            const valarray< _Tp > & ) const
```

Subtract corresponding elements of *v* from slice elements.

operator/=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator/= (
            const valarray< _Tp > & ) const
```

Divide slice elements by corresponding elements of *v*.

operator<<=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator<<= (
            const valarray< _Tp > & ) const
```

Left shift slice elements by corresponding elements of *v*.

operator>>=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator>>= (
            const valarray< _Tp > & ) const
```

Right shift slice elements by corresponding elements of *v*.

operator^=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator^= (
            const valarray< _Tp > & ) const
```

Logical xor slice elements with corresponding elements of *v*.

operator"|=()

```
template<class _Tp>
void std::mask_array< _Tp >::operator|= (
            const valarray< _Tp > & ) const
```

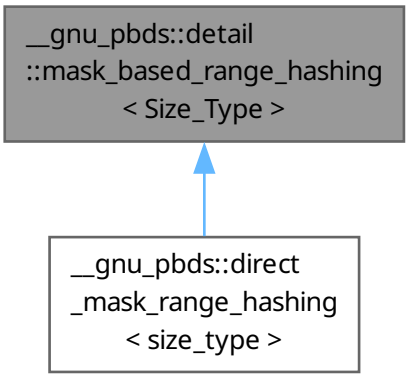
Logical or slice elements with corresponding elements of *v*.
The documentation for this class was generated from the following files:

- valarray
- mask_array.h

6.678 __gnu_pbds::detail::mask_based_range_hashing< Size_Type > Class Template Reference

```
#include <mask_based_range_hashing.hpp>
```

Inheritance diagram for __gnu_pbds::detail::mask_based_range_hashing< Size_Type >:



Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- void **notify_resized** (size_type size)
- size_type **range_hash** (size_type hash) const
- void **swap** (mask_based_range_hashing &other)

6.678.1 Detailed Description

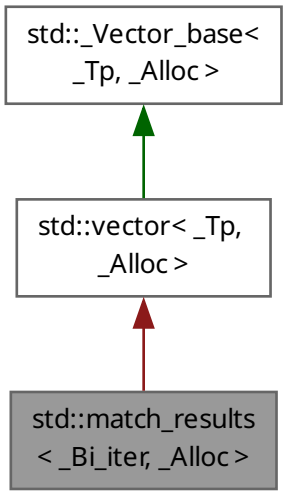
template<typename Size_Type>
class __gnu_pbds::detail::mask_based_range_hashing< Size_Type >

Range hashing policy.
The documentation for this class was generated from the following file:

- mask_based_range_hashing.hpp

6.679 std::match_results< _Bi_iter, _Alloc > Class Template Reference

`#include <regex>`
Inheritance diagram for std::match_results< _Bi_iter, _Alloc >:



Public Types

28.10 Public Types

- typedef sub_match< _Bi_iter > **value_type**
- typedef const value_type & **const_reference**
- typedef value_type & **reference**
- typedef _Base_type::const_iterator **const_iterator**
- typedef const_iterator **iterator**
- typedef __iter_traits::difference_type **difference_type**
- typedef allocator_traits< _Alloc >::size_type **size_type**
- typedef _Alloc **allocator_type**

- typedef __iter_traits::value_type **char_type**
- typedef std::basic_string< char_type > **string_type**

Public Member Functions

- template<typename _Out_iter>
  _Out_iter **format** (_Out_iter __out, const match_results< _Bi_iter, _Alloc >::char_type *__fmt_first, const match_results< _Bi_iter, _Alloc >::char_type *__fmt_last, match_flag_type __flags) const
- bool ready () const noexcept

28.10.1 Construction, Copying, and Destruction

- match_results ()
- match_results (const _Alloc &__a) noexcept
- match_results (const match_results &)=default
- match_results (match_results &&) noexcept=default
- match_results & operator= (const match_results &)=default
- match_results & operator= (match_results &&)=default
- ~match_results ()=default
- match_results (const match_results &__m, const _Alloc &__a)
- match_results (match_results &&__m, const _Alloc &__a) noexcept(noexcept(_Base_type(std::move(__m), ↩ __a)))

28.10.2 Size

- size_type size () const noexcept
- size_type max_size () const noexcept
- bool empty () const noexcept

28.10.4 Element Access

- difference_type length (size_type __sub=0) const
- difference_type position (size_type __sub=0) const
- string_type str (size_type __sub=0) const
- const_reference operator[ ] (size_type __sub) const
- const_reference prefix () const
- const_reference suffix () const
- const_iterator begin () const noexcept
- const_iterator cbegin () const noexcept
- const_iterator end () const noexcept
- const_iterator cend () const noexcept

28.10.5 Formatting

These functions perform formatted substitution of the matched character sequences into their target. The format specifiers and escape sequences accepted by these functions are determined by their `flags` *parameter as documented above.*

- template<typename _Out_iter>
  _Out_iter format (_Out_iter __out, const char_type *__fmt_first, const char_type *__fmt_last, match_flag_type __flags=regex_constants::format_default) const
- template<typename _Out_iter, typename _St, typename _Sa>
  _Out_iter format (_Out_iter __out, const basic_string< char_type, _St, _Sa > &__fmt, match_flag_type __↩ flags=regex_constants::format_default) const
- template<typename _St, typename _Sa>
  basic_string< char_type, _St, _Sa > format (const basic_string< char_type, _St, _Sa > &__fmt, match_flag_type __flags=regex_constants::format_default) const
- string_type format (const char_type *__fmt, match_flag_type __flags=regex_constants::format_default) const

28.10.6 Allocator

- allocator_type get_allocator () const noexcept

28.10.7 Swap

- void swap (match_results &__that) noexcept

Private Member Functions

- constexpr iterator begin () noexcept
- constexpr iterator end () noexcept
- constexpr const_reference operator[ ] (size_type __n) const noexcept
- constexpr reference operator[ ] (size_type __n) noexcept
- constexpr void swap (vector &__x) noexcept

Friends

- template<typename, typename, typename>
  class **regex_iterator**

Related Symbols

(Note that these are not member symbols.)

- template<typename _Bi_iter, typename _Alloc>
  bool operator== (const match_results< _Bi_iter, _Alloc > &__m1, const match_results< _Bi_iter, _Alloc > &_↩
  _m2)
- template<typename _Bi_iter, typename _Alloc>
  void swap (match_results< _Bi_iter, _Alloc > &__lhs, match_results< _Bi_iter, _Alloc > &__rhs) noexcept

6.679.1 Detailed Description

template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
class std::match_results< _Bi_iter, _Alloc >

The results of a match or search operation.
A collection of character sequences representing the result of a regular expression match. Storage for the collection is allocated and freed as necessary by the member functions of class template match_results.
This class satisfies the Sequence requirements, with the exception that only the operations defined for a const-qualified Sequence are supported.
The sub_match object stored at index 0 represents sub-expression 0, i.e. the whole match. In this case the sub_↩
match member matched is always true. The sub_match object stored at index n denotes what matched the marked sub-expression n within the matched expression. If the sub-expression n participated in a regular expression match then the sub_match member matched evaluates to true, and members first and second denote the range of characters [first, second) which formed that match. Otherwise matched is false, and members first and second point to the end of the sequence that was searched.

Since

C++11

6.679.2 Constructor & Destructor Documentation

match_results() [1/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results ()  [inline]
```

Constructs a default match_results container.

Postcondition

size() returns 0 and str() returns an empty string.

match_results() [2/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
            const _Alloc & __a)  [inline], [explicit], [noexcept]
```

Constructs a default match_results container.

Postcondition

size() returns 0 and str() returns an empty string.

match_results() [3/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
            const match_results< _Bi_iter, _Alloc > & )  [default]
```

Copy constructs a match_results.

match_results() [4/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
            match_results< _Bi_iter, _Alloc > && )  [default], [noexcept]
```

Move constructs a match_results.

~match_results()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::~match_results ()  [default]
```

Destroys a match_results object.

match_results() [5/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
            const match_results< _Bi_iter, _Alloc > & __m,
            const _Alloc & __a)  [inline]
```

Constructs a default match_results container.

Postcondition

size() returns 0 and str() returns an empty string.

match_results() [6/6]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
std::match_results< _Bi_iter, _Alloc >::match_results (
            match_results< _Bi_iter, _Alloc > && __m,
            const _Alloc & __a)  [inline], [noexcept]
```

Constructs a default match_results container.

Postcondition

size() returns 0 and str() returns an empty string.

6.679.3 Member Function Documentation

begin()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::begin () const  [inline], [noexcept]
```

Gets an iterator to the start of the sub_match collection.

Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::cbegin(), and std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::operator==().

cbegin()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::cbegin () const  [inline], [noexcept]
```

Gets an iterator to the start of the sub_match collection.

cend()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::cend () const  [inline], [noexcept]
```

Gets an iterator to one-past-the-end of the collection.

empty()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
bool std::match_results< _Bi_iter, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Indicates if the match_results contains no results.

Return values

true	The match_results object is empty.
false	The match_results object is not empty.

Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >::operator==(), std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::prefix(), and std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::suffix().

end()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_iterator std::match_results< _Bi_iter, _Alloc >::end () const  [inline], [noexcept]
```

Gets an iterator to one-past-the-end of the collection.
Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::cend(), and std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::operator==().

format() [1/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _Out_iter, typename _St, typename _Sa>
_Out_iter std::match_results< _Bi_iter, _Alloc >::format (
            _Out_iter __out,
            const basic_string< char_type, _St, _Sa > & __fmt,
            match_flag_type __flags = regex_constants::format_default) const  [inline]
```

Precondition

ready() == true

format() [2/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _Out_iter>
_Out_iter std::match_results< _Bi_iter, _Alloc >::format (
            _Out_iter __out,
            const char_type * __fmt_first,
            const char_type * __fmt_last,
            match_flag_type __flags = regex_constants::format_default) const
```

Precondition

ready() == true

Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::format(), std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::format(), and std::match_results< _BidirectionalIterator, polymorphic↩ _allocator< sub_match< _BidirectionalIterator > > >< const char * >::format().

format() [3/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
template<typename _St, typename _Sa>
basic_string< char_type, _St, _Sa > std::match_results< _Bi_iter, _Alloc >::format (
            const basic_string< char_type, _St, _Sa > & __fmt,
            match_flag_type __flags = regex_constants::format_default) const  [inline]
```

Precondition

ready() == true

format() [4/4]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
string_type std::match_results< _Bi_iter, _Alloc >::format (
            const char_type * __fmt,
            match_flag_type __flags = regex_constants::format_default) const  [inline]
```

Precondition

ready() == true

get_allocator()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
allocator_type std::match_results< _Bi_iter, _Alloc >::get_allocator () const  [inline], [noexcept]
```

Gets a copy of the allocator.

length()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
difference_type std::match_results< _Bi_iter, _Alloc >::length (
            size_type __sub = 0) const  [inline]
```

Gets the length of the indicated submatch.

Parameters

__sub	indicates the submatch.

Precondition

ready() == true

This function returns the length of the indicated submatch, or the length of the entire match if __sub is zero (the default).

max_size()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
size_type std::match_results< _Bi_iter, _Alloc >::max_size () const  [inline], [noexcept]
```

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or mark_count() + 1 if a match was successful. Some matches may be empty.

Returns

the number of matches found.

operator=() [1/2]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
match_results & std::match_results< _Bi_iter, _Alloc >::operator= (
            const match_results< _Bi_iter, _Alloc > & )  [default]
```

Assigns rhs to ∗this.

operator=() [2/2]

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
match_results & std::match_results< _Bi_iter, _Alloc >::operator= (
            match_results< _Bi_iter, _Alloc > && )  [default]
```

Move-assigns rhs to ∗this.

operator[]()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::operator[] (
            size_type __sub) const  [inline]
```

Gets a sub_match reference for the match or submatch.

Parameters

__sub	indicates the submatch.

Precondition

ready() == true

This function gets a reference to the indicated submatch, or the entire match if __sub is zero.
If __sub >= size() then this function returns a sub_match with a special value indicating no submatch.

position()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
difference_type std::match_results< _Bi_iter, _Alloc >::position (
            size_type __sub = 0) const  [inline]
```

Gets the offset of the beginning of the indicated submatch.

Parameters

__sub	indicates the submatch.

Precondition

ready() == true

This function returns the offset from the beginning of the target sequence to the beginning of the submatch, unless the value of __sub is zero (the default), in which case this function returns the offset from the beginning of the target sequence to the beginning of the match.

prefix()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::prefix () const  [inline]
```

Gets a sub_match representing the match prefix.

Precondition

ready() == true

This function gets a reference to a sub_match object representing the part of the target range between the start of the target range and the start of the match.
Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::operator==().

ready()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
bool std::match_results< _Bi_iter, _Alloc >::ready () const  [inline], [noexcept]
```

Indicates if the match_results is ready.

Return values

true	The object has a fully-established result state.
false	The object is not ready.

Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::operator==(), std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::operator[](), std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::prefix(), and std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::suffix().

size()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
size_type std::match_results< _Bi_iter, _Alloc >::size () const  [inline], [noexcept]
```

Gets the number of matches and submatches.

The number of matches for a given regular expression will be either 0 if there was no match or mark_count() + 1 if a match was successful. Some matches may be empty.

Returns

the number of matches found.

Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::operator==(), and std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::operator[]().

str()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
string_type std::match_results< _Bi_iter, _Alloc >::str (
             size_type __sub = 0) const  [inline]
```

Gets the match or submatch converted to a string type.

Parameters

__sub	indicates the submatch.

Precondition

ready() == true

This function gets the submatch (or match, if __sub is zero) extracted from the target range and converted to the associated string type.

suffix()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
const_reference std::match_results< _Bi_iter, _Alloc >::suffix () const  [inline]
```

Gets a sub_match representing the match suffix.

Precondition

ready() == true

This function gets a reference to a sub_match object representing the part of the target range between the end of the match and the end of the target range.
Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::operator==().

swap()

```
template<typename _Bi_iter, typename _Alloc = allocator<sub_match<_Bi_iter> >>
void std::match_results< _Bi_iter, _Alloc >::swap (
            match_results< _Bi_iter, _Alloc > & __that)  [inline], [noexcept]
```

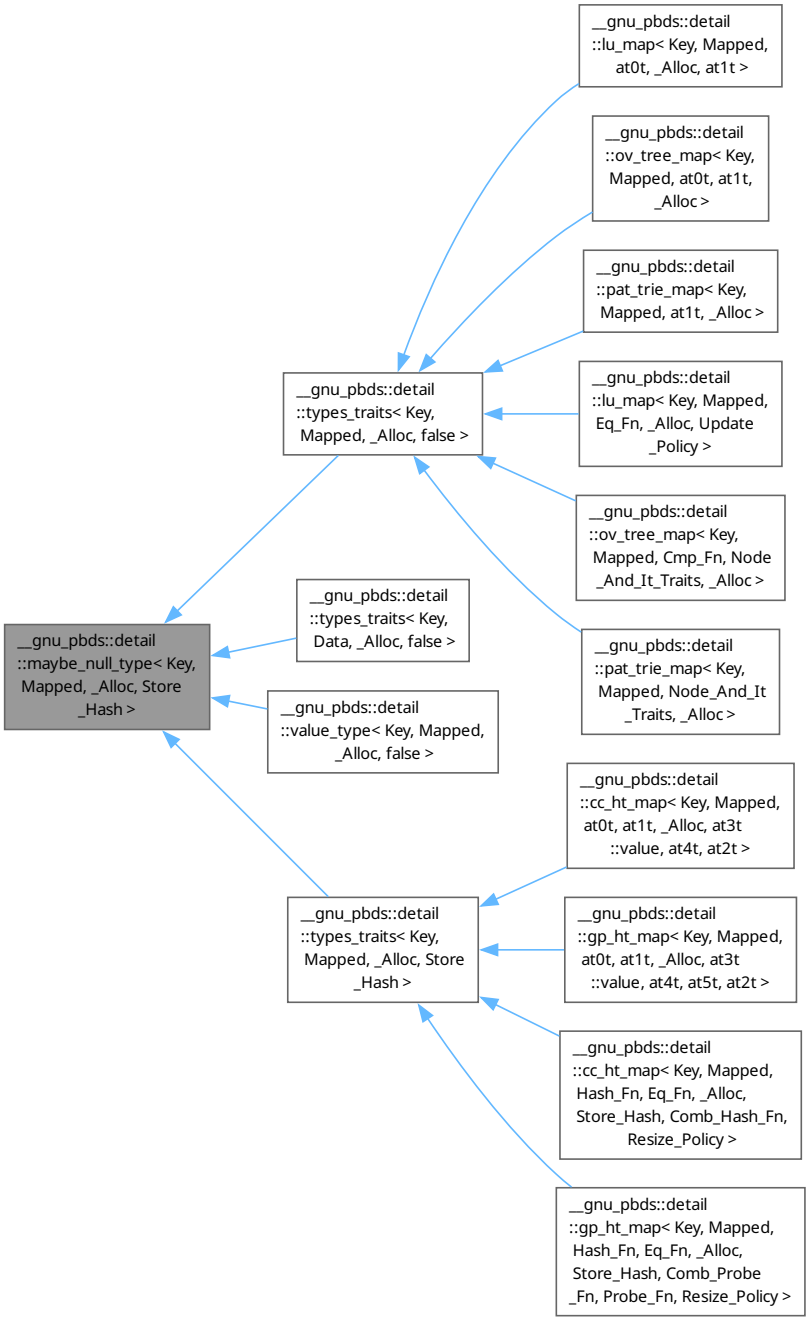
Swaps the contents of two match_results.
Referenced by std::match_results< _BidirectionalIterator, polymorphic_allocator< sub_match< _BidirectionalIterator > > >< const char * >::swap().
The documentation for this class was generated from the following files:

- regex.h
- regex.tcc

6.680 __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference

```
#include <types_traits.hpp>
```

Inheritance diagram for __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >:



6.680.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >

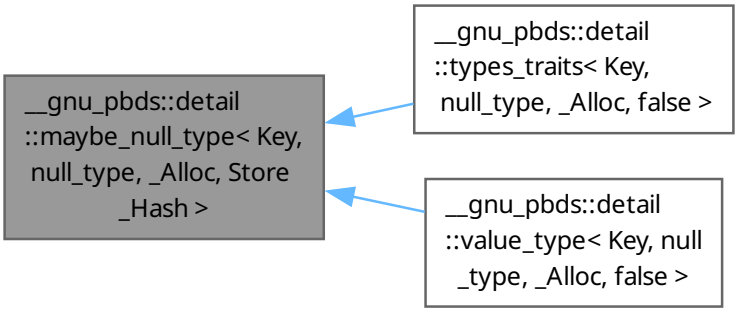
Base class for conditionally defining a static data member.

The documentation for this struct was generated from the following file:

- types_traits.hpp

6.681 __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash > Struct Template Reference

`#include <types_traits.hpp>`
Inheritance diagram for __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >:



Static Public Attributes

- static null_type **s_null_type**

6.681.1 Detailed Description

template<typename Key, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >

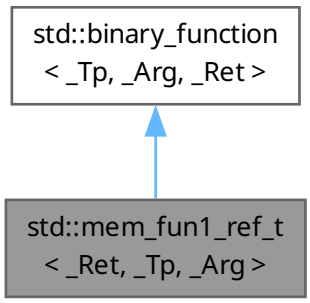
Specialization that defines a static data member of type null_type.
The documentation for this struct was generated from the following file:

- types_traits.hpp

6.682 std::mem_fun1_ref_t< _Ret, _Tp, _Arg > Class Template Reference

`#include <stl_function.h>`

Inheritance diagram for std::mem_fun1_ref_t< _Ret, _Tp, _Arg >:



Public Types

- typedef _Tp first_argument_type
- typedef _Ret result_type
- typedef _Arg second_argument_type

Public Member Functions

- **mem_fun1_ref_t** (_Ret(_Tp::*__pf)(_Arg))
- _Ret **operator()** (_Tp &__r, _Arg __x) const

6.682.1 Detailed Description

template<typename _Ret, typename _Tp, typename _Arg>
class std::mem_fun1_ref_t< _Ret, _Tp, _Arg >

One of the adaptors for member pointers.

6.682.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Arg, _Ret >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef _Ret std::binary_function< _Tp, _Arg, _Ret >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _Arg std::binary_function< _Tp, _Arg, _Ret >::second_argument_type  [inherited]
```

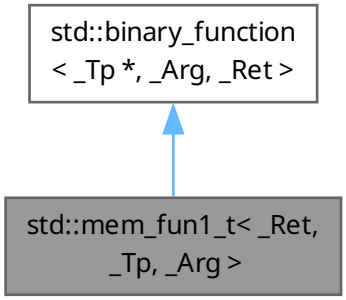
`second_argument_type` is the type of the second argument

The documentation for this class was generated from the following file:

- stl_function.h

6.683 std::mem_fun1_t< _Ret, _Tp, _Arg > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::mem_fun1_t< _Ret, _Tp, _Arg >:



Public Types

- typedef _Tp ∗ first_argument_type
- typedef _Ret result_type
- typedef _Arg second_argument_type

Public Member Functions

- **mem_fun1_t** (_Ret(_Tp::∗__pf)(_Arg))
- _Ret **operator()** (_Tp ∗__p, _Arg __x) const

6.683.1 Detailed Description

template<typename _Ret, typename _Tp, typename _Arg>
class std::mem_fun1_t< _Ret, _Tp, _Arg >

One of the adaptors for member pointers.

6.683.2 Member Typedef Documentation

first_argument_type

`typedef _Tp * std::binary_function< _Tp *, _Arg, _Ret >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef _Ret std::binary_function< _Tp *, _Arg, _Ret >::result_type [inherited]`
`result_type` is the return type

second_argument_type

```
typedef _Arg std::binary_function< _Tp *, _Arg, _Ret >::second_argument_type  [inherited]
```

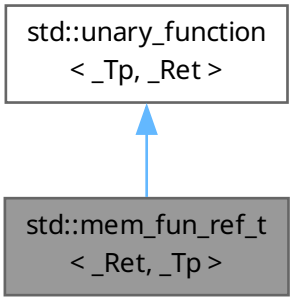
second_argument_type is the type of the second argument
The documentation for this class was generated from the following file:

- stl_function.h

6.684 std::mem_fun_ref_t< _Ret, _Tp > Class Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::mem_fun_ref_t< _Ret, _Tp >:



Public Types

- typedef _Tp argument_type
- typedef _Ret result_type

Public Member Functions

- **mem_fun_ref_t** (_Ret(_Tp::*__pf)())
- _Ret **operator()** (_Tp &__r) const

6.684.1 Detailed Description

template<typename _Ret, typename _Tp>
class std::mem_fun_ref_t< _Ret, _Tp >

One of the adaptors for member pointers.

6.684.2 Member Typedef Documentation

argument_type

```
typedef _Tp std::unary_function< _Tp, _Ret >::argument_type  [inherited]
```

argument_type is the type of the argument

result_type

```
typedef _Ret std::unary_function< _Tp, _Ret >::result_type  [inherited]
```

`result_type` is the return type

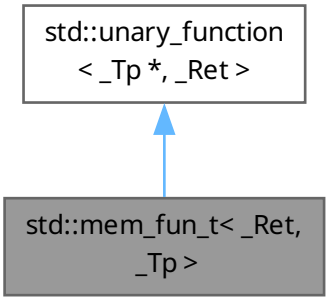
The documentation for this class was generated from the following file:

- stl_function.h

6.685 std::mem_fun_t< _Ret, _Tp > Class Template Reference

`#include <stl_function.h>`

Inheritance diagram for std::mem_fun_t< _Ret, _Tp >:



Public Types

- typedef _Tp * argument_type
- typedef _Ret result_type

Public Member Functions

- **mem_fun_t** (_Ret(_Tp::*__pf)())
- _Ret **operator()** (_Tp *__p) const

6.685.1 Detailed Description

template<typename _Ret, typename _Tp>
class std::mem_fun_t< _Ret, _Tp >

One of the adaptors for member pointers.

6.685.2 Member Typedef Documentation

argument_type

```
typedef _Tp * std::unary_function< _Tp *, _Ret >::argument_type  [inherited]
```

`argument_type` is the type of the argument

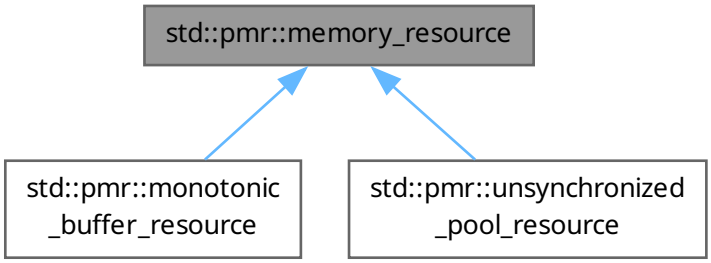
result_type

`typedef _Ret std::unary_function< _Tp *, _Ret >::result_type [inherited]`
`result_type` is the return type
The documentation for this class was generated from the following file:

- stl_function.h

6.686 std::pmr::memory_resource Class Reference

`#include <memory_resource>`
Inheritance diagram for std::pmr::memory_resource:



Public Member Functions

- **memory_resource** (const memory_resource &)=default
- void * **allocate** (size_t __bytes, size_t __alignment=_S_max_align)
- void **deallocate** (void *__p, size_t __bytes, size_t __alignment=_S_max_align)
- bool **is_equal** (const memory_resource &__other) const noexcept
- memory_resource & **operator=** (const memory_resource &)=default

6.686.1 Detailed Description

Class `memory_resource`.

Since

C++17

The documentation for this class was generated from the following file:

- memory_resource.h

6.687 std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > Class Template Reference

`#include <random>`

Public Types

- typedef _UIntType result_type

Public Member Functions

- template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
  mersenne_twister_engine (_Sseq &__q)
- **mersenne_twister_engine** (result_type __sd)
- void discard (unsigned long long __z)
- result_type **operator()** ()
- template<typename _Sseq>
  _If_seed_seq< _Sseq > **seed** (_Sseq &__q)
- void **seed** (result_type __sd=default_seed)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr result_type **default_seed**
- static constexpr result_type **initialization_multiplier**
- static constexpr size_t **mask_bits**
- static constexpr size_t **shift_size**
- static constexpr size_t **state_size**
- static constexpr result_type **tempering_b**
- static constexpr result_type **tempering_c**
- static constexpr result_type **tempering_d**
- static constexpr size_t **tempering_l**
- static constexpr size_t **tempering_s**
- static constexpr size_t **tempering_t**
- static constexpr size_t **tempering_u**
- static constexpr size_t **word_size**
- static constexpr result_type **xor_mask**

Friends

- template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > &__x)
- bool operator== (const mersenne_twister_engine &__lhs, const mersenne_twister_engine &__rhs)
- template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > &__x)

6.687.1 Detailed Description

template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >

A generalized feedback shift register discrete random number generator.

This algorithm avoids multiplication and division and is designed to be friendly to a pipelined architecture. If the parameters are chosen correctly, this generator will produce numbers with a very long period and fairly good apparent entropy, although still not cryptographically strong.
The best way to use this generator is with the predefined mt19937 class.
This algorithm was originally invented by Makoto Matsumoto and Takuji Nishimura.

Template Parameters

_↩ _w	Word size, the number of bits in each element of the state vector.
_↩ _n	The degree of recursion.
_↩ _m	The period parameter.
_↩ _r	The separation point bit index.
_↩ _a	The last row of the twist matrix.
_↩ _u	The first right-shift tempering matrix parameter.
_↩ _d	The first right-shift tempering matrix mask.
_↩ _s	The first left-shift tempering matrix parameter.
_↩ _b	The first left-shift tempering matrix mask.
_↩ _t	The second left-shift tempering matrix parameter.
_↩ _c	The second left-shift tempering matrix mask.
_↩ _l	The second right-shift tempering matrix parameter.
_↩ _f	Initialization multiplier.

Since

C++11

6.687.2 Member Typedef Documentation

result_type

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size↩
_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f>
typedef _UIntType std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __↩
s, __b, __t, __c, __l, __f >::result_type
```

The type of the generated random value.

6.687.3 Constructor & Destructor Documentation

mersenne_twister_engine()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size↩
_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, ↩
__l, __f >::mersenne_twister_engine (
            _Sseq & __q)  [inline], [explicit]
```

Constructs a mersenne_twister_engine random number generator engine seeded from the seed sequence __q.

Parameters

_↩ _q	the seed sequence.

6.687.4 Member Function Documentation

discard()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size↩
_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f>
void std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, ↩
__c, __l, __f >::discard (
            unsigned long long __z)
```

Discard a sequence of random numbers.

max()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size↩
_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType
__f>
constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d,
__s, __b, __t, __c, __l, __f >::max ()  [inline], [static], [constexpr]
```

Gets the largest possible value in the output range.

min()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
constexpr result_type std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >::min () [inline], [static], [constexpr]
```

Gets the smallest possible value in the output range.

6.687.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > & __x) [friend]
```

Inserts the current state of a % mersenne_twister_engine random number generator engine __x into the output stream __os.

Parameters

__os	An output stream.
__x	A % mersenne_twister_engine random number generator engine.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
bool operator== (
            const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, _b, __t, __c, __l, __f > & __lhs,
            const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, _b, __t, __c, __l, __f > & __rhs) [friend]
```

Compares two % mersenne_twister_engine random number generator objects of the same type for equality.

Parameters

__lhs	A % mersenne_twister_engine random number generator object.
__rhs	Another % mersenne_twister_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

operator>>

```
template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f>
template<typename _UIntType1, size_t __w1, size_t __n1, size_t __m1, size_t __r1, _UIntType1 __a1, size_t __u1, _UIntType1 __d1, size_t __s1, _UIntType1 __b1, size_t __t1, _UIntType1 __c1, size_t __l1, _UIntType1 __f1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::mersenne_twister_engine< _UIntType1, __w1, __n1, __m1, __r1, __a1, __u1, __d1, __s1, __b1, __t1, __c1, __l1, __f1 > & __x)  [friend]
```

Extracts the current state of a % mersenne_twister_engine random number generator engine __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A % mersenne_twister_engine random number generator engine.

Returns

The input stream with the state of __x extracted or in an error state.

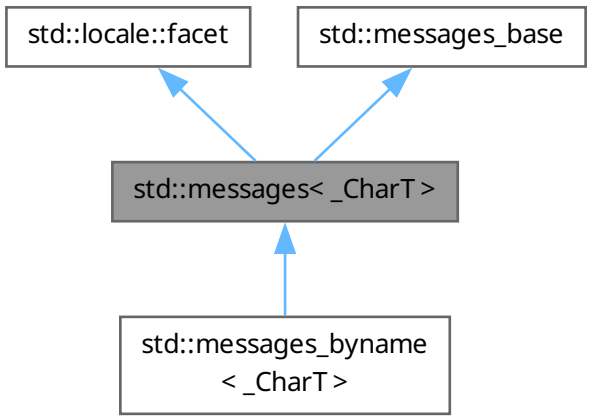
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.688 std::messages< _CharT > Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::messages< _CharT >:



Public Types

- typedef int **catalog**

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- messages (__c_locale __cloc, const char *__s, size_t __refs=0)
- messages (size_t __refs=0)
- void **close** (catalog __c) const
- string_type **get** (catalog __c, int __set, int __msgid, const string_type &__s) const
- catalog **open** (const basic_string< char > &, const locale &, const char *) const
- catalog **open** (const basic_string< char > &__s, const locale &__loc) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~messages ()
- string_type **_M_convert_from_char** (char *) const
- char * **_M_convert_to_char** (const string_type &__msg) const
- virtual void **do_close** (catalog) const
- void **do_close** (catalog) const
- void **do_close** (catalog) const
- string do_get (catalog, int, int, const string &) const
- virtual string_type **do_get** (catalog, int, int, const string_type &__dfault) const

- wstring **do_get** (catalog, int, int, const wstring &) const
- virtual catalog **do_open** (const basic_string< char > &, const locale &) const
- messages< wchar_t >::catalog **do_open** (const basic_string< char > &, const locale &) const
- messages< char >::catalog **do_open** (const basic_string< char > &, const locale &) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_messages**
- const char * **_M_name_messages**

6.688.1 Detailed Description

template<typename _CharT>
class std::messages< _CharT >

Primary class template messages.
This facet encapsulates the code to retrieve messages from message catalogs. The only thing defined by the standard for this facet is the interface. All underlying functionality is implementation-defined.
This library currently implements 3 versions of the message facet. The first version (gnu) is a wrapper around gettext, provided by libintl. The second version (ieee) is a wrapper around catgets. The final version (default) does no actual translation. These implementations are only provided for char and wchar_t instantiations.
The messages template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the messages facet.

6.688.2 Member Typedef Documentation

char_type

```
template<typename _CharT>
typedef _CharT std::messages< _CharT >::char_type
```

Public typedefs.

string_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::messages< _CharT >::string_type
```

Public typedefs.

6.688.3 Constructor & Destructor Documentation

messages() [1/2]

```
template<typename _CharT>
std::messages< _CharT >::messages (
            size_t __refs = 0)  [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().

messages() [2/2]

```
template<typename _CharT>
std::messages< _CharT >::messages (
            __c_locale __cloc,
            const char * __s,
            size_t __refs = 0)  [explicit]
```

Internal constructor. Not for general use.
This is a constructor for use by the library itself to set up new locales.

Parameters

__cloc	The C locale.
__s	The name of a locale.
__refs	Refcount to pass to the base class.

References std::locale::facet::facet().

~messages()

```
template<typename _CharT>
std::messages< _CharT >::~messages ()  [protected], [virtual]
```

Destructor.

6.688.4 Member Function Documentation

do_get()

```
string std::messages< char >::do_get (
            catalog ,
            int ,
            int ,
            const string & ) const  [protected]
```

Specializations for required instantiations.

6.688.5 Member Data Documentation

id

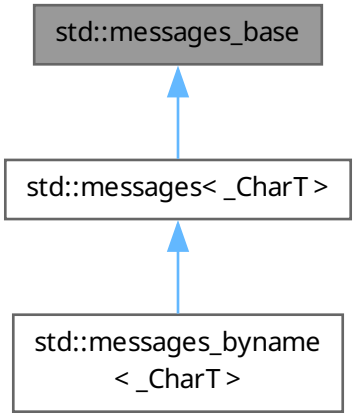
```
template<typename _CharT>
locale::id std::messages< _CharT >::id  [static]
```

Numpunct facet id.
The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- messages_members.h

6.689 std::messages_base Struct Reference

`#include <locale_facets_nonio.h>`
Inheritance diagram for std::messages_base:



Public Types

- typedef int **catalog**

6.689.1 Detailed Description

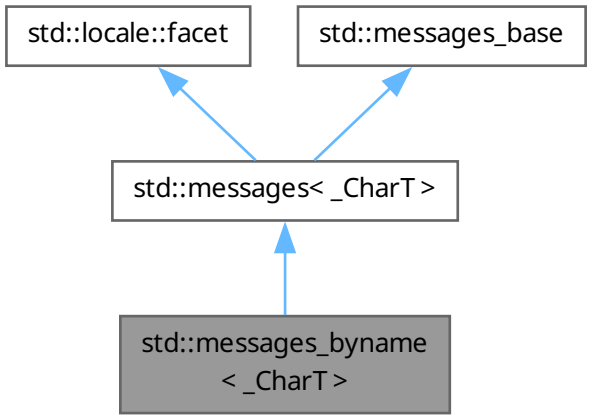
Messages facet base class providing catalog typedef.
The documentation for this struct was generated from the following file:

- locale_facets_nonio.h

6.690 std::messages_byname< _CharT > Class Template Reference

`#include <locale_facets_nonio.h>`

Inheritance diagram for std::messages_byname< _CharT >:



Public Types

- typedef int **catalog**
- typedef _CharT **char_type**
- typedef basic_string< _CharT > **string_type**

Public Member Functions

- **messages_byname** (const char *__s, size_t __refs=0)
- **messages_byname** (const string &__s, size_t __refs=0)
- void **close** (catalog __c) const
- string_type **get** (catalog __c, int __set, int __msgid, const string_type &__s) const
- catalog **open** (const basic_string< char > &, const locale &, const char *) const
- catalog **open** (const basic_string< char > &__s, const locale &__loc) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- string_type **_M_convert_from_char** (char *) const
- char * **_M_convert_to_char** (const string_type &__msg) const
- virtual void **do_close** (catalog) const
- void **do_close** (catalog) const
- void **do_close** (catalog) const
- string do_get (catalog, int, int, const string &) const
- virtual string_type **do_get** (catalog, int, int, const string_type &__dfault) const
- wstring **do_get** (catalog, int, int, const wstring &) const
- virtual catalog **do_open** (const basic_string< char > &, const locale &) const
- messages< wchar_t >::catalog **do_open** (const basic_string< char > &, const locale &) const
- messages< char >::catalog **do_open** (const basic_string< char > &, const locale &) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __c_locale **_M_c_locale_messages**
- const char * **_M_name_messages**

6.690.1 Detailed Description

template<typename _CharT>
class std::messages_byname< _CharT >

class messages_byname [22.2.7.2].

6.690.2 Member Function Documentation

do_get()

```
string std::messages< char >::do_get (
             catalog ,
             int ,
             int ,
             const string & ) const  [protected], [inherited]
```

Specializations for required instantiations.

6.690.3 Member Data Documentation

id

```
template<typename _CharT>
locale::id std::messages< _CharT >::id  [static], [inherited]
```

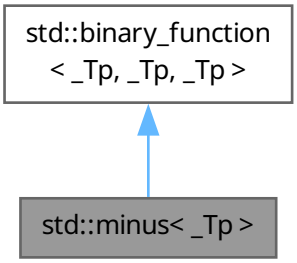
Numpunct facet id.
The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- messages_members.h

6.691 std::minus< _Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::minus< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

6.691.1 Detailed Description

template<typename _Tp>
struct std::minus< _Tp >

One of the math functors.

6.691.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

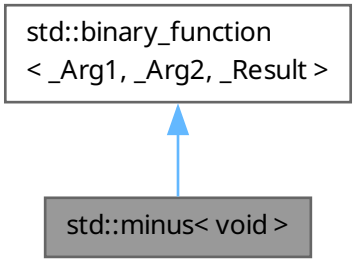
```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.692 std::minus< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::minus< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef void result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) - std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) - std::forward< _Up >(__u))

6.692.1 Detailed Description

One of the math functors.

6.692.2 Member Typedef Documentation

first_argument_type

`typedef void std::binary_function< void, void, void >::first_argument_type`
`first_argument_type` is the type of the first argument

result_type

`typedef void std::binary_function< void, void, void >::result_type`
`result_type` is the return type

second_argument_type

`typedef void std::binary_function< void, void, void >::second_argument_type`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.693 __gnu_pbds::detail::mod_based_range_hashing< Size_Type > Class Template Reference

`#include <mod_based_range_hashing.hpp>`

Protected Types

- typedef Size_Type **size_type**

Protected Member Functions

- void **notify_resized** (size_type s)
- size_type **range_hash** (size_type s) const
- void **swap** (mod_based_range_hashing &other)

6.693.1 Detailed Description

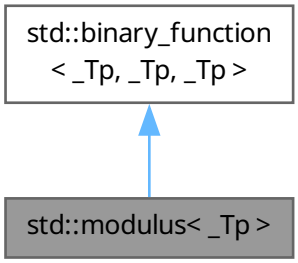
template<typename Size_Type>
class __gnu_pbds::detail::mod_based_range_hashing< Size_Type >

Mod based range hashing.
The documentation for this class was generated from the following file:

- mod_based_range_hashing.hpp

6.694 std::modulus< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::modulus< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

6.694.1 Detailed Description

template<**typename _Tp**>
struct std::modulus< **_Tp** >

One of the math functors.

6.694.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type [inherited]`
`result_type` is the return type

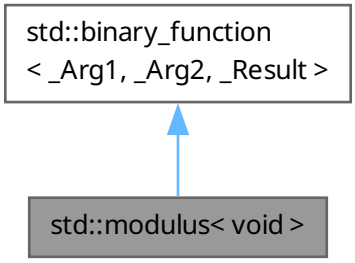
second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.695 std::modulus< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::modulus< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**

- typedef void result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) % std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) % std::forward< _Up >(__u))

6.695.1 Detailed Description

One of the math functors.

6.695.2 Member Typedef Documentation

first_argument_type

```
typedef void std::binary_function< void, void, void >::first_argument_type
```

`first_argument_type` is the type of the first argument

result_type

```
typedef void std::binary_function< void, void, void >::result_type
```

`result_type` is the return type

second_argument_type

```
typedef void std::binary_function< void, void, void >::second_argument_type
```

`second_argument_type` is the type of the second argument

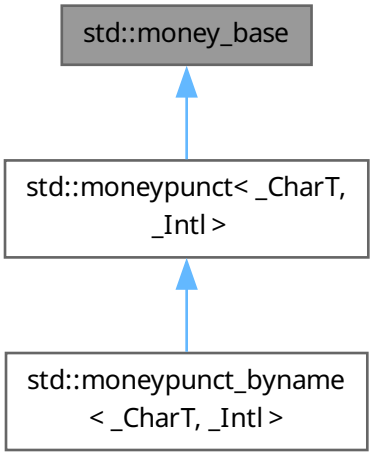
The documentation for this struct was generated from the following file:

- stl_function.h

6.696 std::money_base Class Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::money_base:



Public Types

- enum { **_S_minus** , **_S_zero** , **_S_end** }
- enum **part** {
  none , **space** , **symbol** , **sign** ,
  value }

Static Public Member Functions

- static pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char * **_S_atoms**
- static const pattern **_S_default_pattern**

6.696.1 Detailed Description

Money format ordering data.
This class contains an ordered array of 4 fields to represent the pattern for formatting a money amount. Each field may contain one entry from the part enum. symbol, sign, and value must be present and the remaining field must contain either none or space.

See also

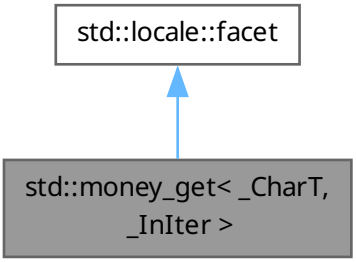
moneypunct::pos_format() and moneypunct::neg_format() for details of how these fields are interpreted.

The documentation for this class was generated from the following file:

- locale_facets_nonio.h

6.697 std::money_get< _CharT, _InIter > Class Template Reference

`#include <locale_facets_nonio.h>`
Inheritance diagram for std::money_get< _CharT, _InIter >:



Public Types

- typedef _CharT char_type
- typedef _InIter iter_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- money_get (size_t __refs=0)
- template<bool _Intl>
  _GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11 _InIter **_M_extract** (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, string &__units) const
- iter_type get (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, long double &__units) const
- iter_type get (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, string_type &__digits) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~money_get ()
- template<bool _Intl>
  iter_type **_M_extract** (iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, string &__digits) const
- virtual iter_type do_get (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, long double &__units) const
- virtual iter_type do_get (iter_type __s, iter_type __end, bool __intl, ios_base &__io, ios_base::iostate &__err, string_type &__digits) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.697.1 Detailed Description

template<typename _CharT, typename _InIter>
class std::money_get< _CharT, _InIter >

Primary class template money_get.
This facet encapsulates the code to parse and return a monetary amount from a string.
The money_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the money_get facet.

6.697.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _InIter>
typedef _CharT std::money_get< _CharT, _InIter >::char_type
```

Public typedefs.

iter_type

```
template<typename _CharT, typename _InIter>
typedef _InIter std::money_get< _CharT, _InIter >::iter_type
```

Public typedefs.

string_type

```
template<typename _CharT, typename _InIter>
typedef basic_string<_CharT> std::money_get< _CharT, _InIter >::string_type
```

Public typedefs.

6.697.3 Constructor & Destructor Documentation

money_get()

```
template<typename _CharT, typename _InIter>
std::money_get< _CharT, _InIter >::money_get (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().

~money_get()

```
template<typename _CharT, typename _InIter>
virtual std::money_get< _CharT, _InIter >::~money_get ()  [inline], [protected], [virtual]
```

Destructor.

6.697.4 Member Function Documentation

do_get() [1/2]

```
template<typename _CharT, typename _InIter>
_InIter std::money_get< _CharT, _InIter >::do_get (
            iter_type __s,
            iter_type __end,
            bool __intl,
            ios_base & __io,
            ios_base::iostate & __err,
            long double & __units) const  [protected], [virtual]
```

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

get() for details.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str().

Referenced by get(), and get().

do_get() [2/2]

```
template<typename _CharT, typename _InIter>
_InIter std::money_get< _CharT, _InIter >::do_get (
            iter_type __s,
            iter_type __end,
            bool __intl,
            ios_base & __io,
            ios_base::iostate & __err,
            string_type & __digits) const  [protected], [virtual]
```

Read and parse a monetary value.

This function reads and parses characters representing a monetary value. This function is a hook for derived classes to change the value returned.

See also

get() for details.

References std::ios_base::_M_getloc(), std::use_facet(), and std::__ctype_abstract_base< _CharT >::widen().

get() [1/2]

```
template<typename _CharT, typename _InIter>
iter_type std::money_get< _CharT, _InIter >::get (
            iter_type __s,
            iter_type __end,
            bool __intl,
            ios_base & __io,
```

```
            ios_base::iostate & __err,
            long double & __units) const  [inline]
```

Read and parse a monetary value.

This function reads characters from *__s*, interprets them as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and returns the result in *units* as an integral value moneypunct::frac_digits() * the actual amount. For example, the string $10.01 in a US locale would store 1001 in *units*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets err=(err|io.failbit). If the stream is consumed before finishing parsing, sets err=(err|io.failbit|io.eofbit). *units* is unchanged if parsing fails.

This function works by returning the result of do_get().

Parameters

__s	Start of characters to parse.
__end	End of characters to parse.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__err	Error field to set if parsing fails.
__units	Place to store result of parsing.

Returns

Iterator referencing first character beyond valid money amount.

References do_get().

get() [2/2]

```
template<typename _CharT, typename _InIter>
iter_type std::money_get< _CharT, _InIter >::get (
            iter_type __s,
            iter_type __end,
            bool __intl,
            ios_base & __io,
            ios_base::iostate & __err,
            string_type & __digits) const  [inline]
```

Read and parse a monetary value.

This function reads characters from *__s*, interprets them as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and returns the result in *digits*. For example, the string $10.01 in a US locale would store `1001` in *digits*.

Any characters not part of a valid money amount are not consumed.

If a money value cannot be parsed from the input stream, sets err=(err|io.failbit). If the stream is consumed before finishing parsing, sets err=(err|io.failbit|io.eofbit).

This function works by returning the result of do_get().

Parameters

__s	Start of characters to parse.

__end	End of characters to parse.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__err	Error field to set if parsing fails.
__digits	Place to store result of parsing.

Returns

Iterator referencing first character beyond valid money amount.

References do_get().

6.697.5 Member Data Documentation

id

```
template<typename _CharT, typename _InIter>
locale::id std::money_get< _CharT, _InIter >::id  [static]
```

Numpunct facet id.

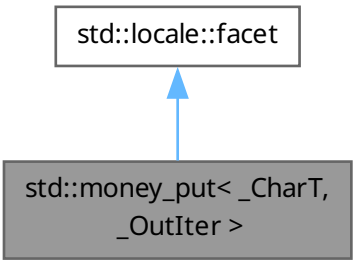
The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- locale_facets_nonio.tcc

6.698 std::money_put< _CharT, _OutIter > Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::money_put< _CharT, _OutIter >:



Public Types

- typedef _CharT char_type
- typedef _OutIter iter_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- money_put (size_t __refs=0)
- template<bool _Intl>
  _OutIter **_M_insert** (iter_type __s, ios_base &__io, char_type __fill, const string_type &__digits) const
- iter_type put (iter_type __s, bool __intl, ios_base &__io, char_type __fill, const string_type &__digits) const
- iter_type put (iter_type __s, bool __intl, ios_base &__io, char_type __fill, long double __units) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~money_put ()
- template<bool _Intl>
  iter_type **_M_insert** (iter_type __s, ios_base &__io, char_type __fill, const string_type &__digits) const
- virtual iter_type do_put (iter_type __s, bool __intl, ios_base &__io, char_type __fill, const string_type &__digits) const
- virtual iter_type do_put (iter_type __s, bool __intl, ios_base &__io, char_type __fill, long double __units) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.698.1 Detailed Description

template<typename _CharT, typename _OutIter>
class std::money_put< _CharT, _OutIter >

Primary class template money_put.
This facet encapsulates the code to format and output a monetary amount.
The money_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the money_put facet.

6.698.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _OutIter>
typedef _CharT std::money_put< _CharT, _OutIter >::char_type
```

Public typedefs.

iter_type

```
template<typename _CharT, typename _OutIter>
typedef _OutIter std::money_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

string_type

```
template<typename _CharT, typename _OutIter>
typedef basic_string<_CharT> std::money_put< _CharT, _OutIter >::string_type
```

Public typedefs.

6.698.3 Constructor & Destructor Documentation

money_put()

```
template<typename _CharT, typename _OutIter>
std::money_put< _CharT, _OutIter >::money_put (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.

This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().

~money_put()

```
template<typename _CharT, typename _OutIter>
virtual std::money_put< _CharT, _OutIter >::~money_put ()  [inline], [protected], [virtual]
```

Destructor.

6.698.4 Member Function Documentation

do_put() [1/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::money_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            bool __intl,
            ios_base & __io,
            char_type __fill,
            const string_type & __digits) const  [protected], [virtual]
```

Format and output a monetary value.

This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *__s*. For example, the string `1001` in a US locale would write `$10.01` to *__s*. This function is a hook for derived classes to change the value returned.

See also

put().

Parameters

__s	The stream to write to.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.

__io	Source of facets and io state.
__fill	char_type to use for padding.
__digits	Place to store result of parsing.

Returns

Iterator after writing.

do_put() [2/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::money_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            bool __intl,
            ios_base & __io,
            char_type __fill,
            long double __units) const  [protected], [virtual]
```

Format and output a monetary value.
This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *__s*. For example, the value 1001 in a US locale would write `$10.01` to *__s*.
This function is a hook for derived classes to change the value returned.

See also

put().

Parameters

__s	The stream to write to.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__fill	char_type to use for padding.
__units	Place to store result of parsing.

Returns

Iterator after writing.

References std::ios_base::getloc(), std::use_facet(), and std::__ctype_abstract_base< _CharT >::widen().
Referenced by put(), and put().

put() [1/2]

```
template<typename _CharT, typename _OutIter>
iter_type std::money_put< _CharT, _OutIter >::put (
```

```
            iter_type __s,
            bool __intl,
            ios_base & __io,
            char_type __fill,
            const string_type & __digits) const  [inline]
```

Format and output a monetary value.
This function formats *digits* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *__s*. For example, the string `1001` in a US locale would write `$10.01` to *__s*. This function works by returning the result of do_put().

Parameters

__s	The stream to write to.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__fill	char_type to use for padding.
__digits	Place to store result of parsing.

Returns

Iterator after writing.

References do_put().

put() [2/2]

```
template<typename _CharT, typename _OutIter>
iter_type std::money_put< _CharT, _OutIter >::put (
            iter_type __s,
            bool __intl,
            ios_base & __io,
            char_type __fill,
            long double __units) const  [inline]
```

Format and output a monetary value.
This function formats *units* as a monetary value according to moneypunct and ctype facets retrieved from io.getloc(), and writes the resulting characters to *__s*. For example, the value 1001 in a US locale would write `$10.01` to *__s*. This function works by returning the result of do_put().

Parameters

__s	The stream to write to.
__intl	Parameter to use_facet<moneypunct<CharT,intl> >.
__io	Source of facets and io state.
__fill	char_type to use for padding.

__units	Place to store result of parsing.

Returns

Iterator after writing.

References do_put().

6.698.5 Member Data Documentation

id

```
template<typename _CharT, typename _OutIter>
locale::id std::money_put< _CharT, _OutIter >::id  [static]
```

Numpunct facet id.

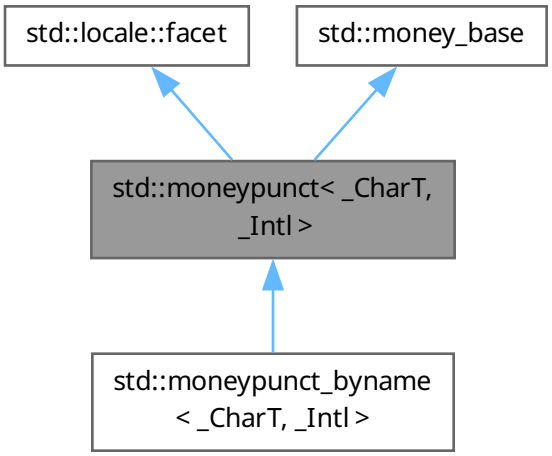
The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- locale_facets_nonio.tcc

6.699 std::moneypunct< _CharT, _Intl > Class Template Reference

`#include <locale_facets_nonio.h>`

Inheritance diagram for std::moneypunct< _CharT, _Intl >:



Public Types

- enum { **_S_minus** , **_S_zero** , **_S_end** }
- typedef __moneypunct_cache< _CharT, _Intl > **__cache_type**
- enum **part** {
  none , **space** , **symbol** , **sign** ,
  value }

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- moneypunct (__c_locale __cloc, const char *__s, size_t __refs=0)
- moneypunct (__cache_type *__cache, size_t __refs=0)
- moneypunct (size_t __refs=0)
- string_type curr_symbol () const
- char_type decimal_point () const
- int frac_digits () const
- string grouping () const
- string_type negative_sign () const
- string_type positive_sign () const
- char_type thousands_sep () const

- pattern pos_format () const
- pattern neg_format () const

Static Public Member Functions

- static pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char * **_S_atoms**
- static const pattern **_S_default_pattern**
- static locale::id id
- static const bool intl

Protected Member Functions

- virtual ~moneypunct ()
- void **_M_initialize_moneypunct** (__c_locale __cloc=0, const char *__name=0)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- virtual string_type do_curr_symbol () const
- virtual char_type do_decimal_point () const
- virtual int do_frac_digits () const
- virtual string do_grouping () const
- virtual pattern do_neg_format () const
- virtual string_type do_negative_sign () const
- virtual pattern do_pos_format () const
- virtual string_type do_positive_sign () const
- virtual char_type do_thousands_sep () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.699.1 Detailed Description

template<typename _CharT, bool _Intl>
class std::moneypunct< _CharT, _Intl >

Primary class template moneypunct.
This facet encapsulates the punctuation, grouping and other formatting features of money amount string representations.

6.699.2 Member Typedef Documentation

char_type

```
template<typename _CharT, bool _Intl>
typedef _CharT std::moneypunct< _CharT, _Intl >::char_type
```

Public typedefs.

string_type

```
template<typename _CharT, bool _Intl>
typedef basic_string<_CharT> std::moneypunct< _CharT, _Intl >::string_type
```

Public typedefs.

6.699.3 Constructor & Destructor Documentation

moneypunct() [1/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct< _CharT, _Intl >::moneypunct (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().

moneypunct() [2/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct< _CharT, _Intl >::moneypunct (
            __cache_type * __cache,
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is an internal constructor.

Parameters

__cache	Cache for optimization.
__refs	Passed to the base facet class.

References std::locale::facet::facet().

moneypunct() [3/3]

```
template<typename _CharT, bool _Intl>
std::moneypunct< _CharT, _Intl >::moneypunct (
            __c_locale __cloc,
            const char * __s,
            size_t __refs = 0)  [inline], [explicit]
```

Internal constructor. Not for general use.
This is a constructor for use by the library itself to set up new locales.

Parameters

__cloc	The C locale.
__s	The name of a locale.
__refs	Passed to the base facet class.

References std::locale::facet::facet().

~moneypunct()

```
template<typename _CharT, bool _Intl>
virtual std::moneypunct< _CharT, _Intl >::~moneypunct ()  [protected], [virtual]
```

Destructor.

6.699.4 Member Function Documentation

curr_symbol()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::curr_symbol () const  [inline]
```

Return currency symbol string.
This function returns a string_type to use as a currency symbol. It does so by returning returning moneypunct<char↩ _type>::do_curr_symbol().

Returns

string_type representing a currency symbol.

References do_curr_symbol().

decimal_point()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::decimal_point () const  [inline]
```

Return decimal point character.
This function returns a char_type to use as a decimal point. It does so by returning returning moneypunct<char_↩ type>::do_decimal_point().

Returns

char_type representing a decimal point.

References do_decimal_point().

do_curr_symbol()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_curr_symbol () const  [inline], [protected],
[virtual]
```

Return currency symbol string.

This function returns a string_type to use as a currency symbol. This function is a hook for derived classes to change the value returned.

See also

curr_symbol() for details.

Returns

string_type representing a currency symbol.

Referenced by curr_symbol().

do_decimal_point()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_decimal_point () const  [inline], [protected],
[virtual]
```

Return decimal point character.

Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Referenced by decimal_point().

do_frac_digits()

```
template<typename _CharT, bool _Intl>
virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits () const  [inline], [protected],
[virtual]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

See also

frac_digits() for details.

Returns

Number of digits in amount fraction.

Referenced by frac_digits().

do_grouping()

```
template<typename _CharT, bool _Intl>
virtual string std::moneypunct< _CharT, _Intl >::do_grouping () const  [inline], [protected],
[virtual]
```

Return grouping specification.
Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

grouping() for details.

Returns

String representing grouping specification.

Referenced by grouping().

do_neg_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format () const  [inline], [protected],
[virtual]
```

Return pattern for money values.
This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See also

neg_format() for details.

Returns

Pattern for money values.

Referenced by neg_format().

do_negative_sign()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_negative_sign () const  [inline], [protected],
[virtual]
```

Return negative sign string.
This function returns a string_type to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

See also

negative_sign() for details.

Returns

string_type representing a negative sign.

Referenced by negative_sign().

do_pos_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format () const  [inline], [protected],
[virtual]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

See also

pos_format() for details.

Returns

Pattern for money values.

Referenced by pos_format().

do_positive_sign()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_positive_sign () const  [inline], [protected],
[virtual]
```

Return positive sign string.

This function returns a string_type to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

See also

positive_sign() for details.

Returns

string_type representing a positive sign.

Referenced by positive_sign().

do_thousands_sep()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_thousands_sep () const  [inline], [protected],
[virtual]
```

Return thousands separator character.

Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Referenced by thousands_sep().

frac_digits()

```
template<typename _CharT, bool _Intl>
int std::moneypunct< _CharT, _Intl >::frac_digits () const  [inline]
```

Return number of digits in fraction.

This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning moneypunct<char_type>::do_frac_digits().

The fractional part of a money amount is optional. But if it is present, there must be frac_digits() digits.

Returns

Number of digits in amount fraction.

References do_frac_digits().

grouping()

```
template<typename _CharT, bool _Intl>
string std::moneypunct< _CharT, _Intl >::grouping () const  [inline]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.

Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling moneypunct<char_type>::do_grouping().

Returns

string representing grouping specification.

References do_grouping().

neg_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::neg_format () const  [inline]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char_type>::do_pos_format() or moneypunct<char_type>::do_neg_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr_symbol() may be present. The sign field indicates that the value of positive_sign() or negative_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos_format() pattern {symbol,sign,value,none}, curr_symbol() == '$' positive_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

References do_neg_format().

negative_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::negative_sign () const  [inline]
```

Return negative sign string.

This function returns a string_type to use as a sign for negative amounts. It does so by returning returning moneypunct<char_type>::do_negative_sign().

If the return value contains more than one character, the first character appears in the position indicated by neg_format() and the remainder appear at the end of the formatted string.

Returns

string_type representing a negative sign.

References do_negative_sign().

pos_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::pos_format () const  [inline]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char_type>::do_pos_format() or moneypunct<char_type>::do_neg_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field.

The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr_symbol() may be present. The sign field indicates that the value of positive_sign() or negative_sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos_format() pattern {symbol,sign,value,none}, curr_symbol() == '$' positive_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

References do_pos_format().

positive_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::positive_sign () const  [inline]
```

Return positive sign string.

This function returns a string_type to use as a sign for positive amounts. It does so by returning returning moneypunct<char_type>::do_positive_sign().

If the return value contains more than one character, the first character appears in the position indicated by pos_format() and the remainder appear at the end of the formatted string.

Returns

string_type representing a positive sign.

References do_positive_sign().

thousands_sep()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::thousands_sep () const  [inline]
```

Return thousands separator character.

This function returns a char_type to use as a thousands separator. It does so by returning returning moneypunct<char_type>::do_thousands_sep().

Returns

char_type representing a thousands separator.

References do_thousands_sep().

6.699.5 Member Data Documentation

id

```
template<typename _CharT, bool _Intl>
locale::id std::moneypunct< _CharT, _Intl >::id  [static]
```

Numpunct facet id.

intl

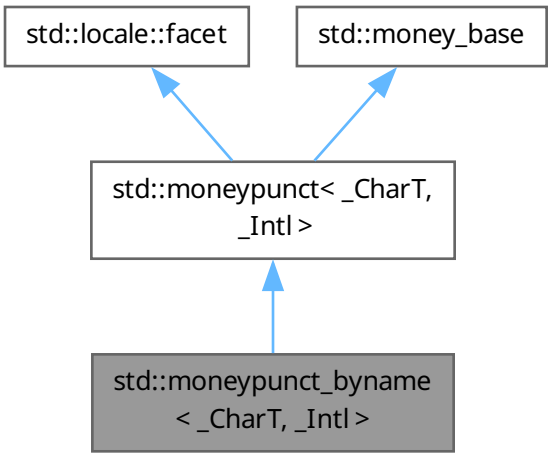
```
template<typename _CharT, bool _Intl>
const bool std::moneypunct< _CharT, _Intl >::intl  [static]
```

This value is provided by the standard, but no reason for its existence.
The documentation for this class was generated from the following file:

- locale_facets_nonio.h

6.700 std::moneypunct_byname< _CharT, _Intl > Class Template Reference

`#include <locale_facets_nonio.h>`
Inheritance diagram for std::moneypunct_byname< _CharT, _Intl >:



Public Types

- enum { **_S_minus** , **_S_zero** , **_S_end** }
- typedef __moneypunct_cache< _CharT, _Intl > **__cache_type**
- typedef _CharT **char_type**
- enum **part** {
  none , **space** , **symbol** , **sign** ,
  value }
- typedef basic_string< _CharT > **string_type**

Public Member Functions

- **moneypunct_byname** (const char *__s, size_t __refs=0)
- **moneypunct_byname** (const string &__s, size_t __refs=0)
- string_type curr_symbol () const
- char_type decimal_point () const
- int frac_digits () const
- string grouping () const
- string_type negative_sign () const
- string_type positive_sign () const
- char_type thousands_sep () const

- pattern pos_format () const
- pattern neg_format () const

Static Public Member Functions

- static pattern **_S_construct_pattern** (char __precedes, char __space, char __posn) throw ()

Static Public Attributes

- static const char * **_S_atoms**
- static const pattern **_S_default_pattern**
- static locale::id id
- static const bool **intl**

Protected Member Functions

- void **_M_initialize_moneypunct** (__c_locale __cloc=0, const char *__name=0)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- void **_M_initialize_moneypunct** (__c_locale, const char *)
- virtual string_type do_curr_symbol () const
- virtual char_type do_decimal_point () const
- virtual int do_frac_digits () const
- virtual string do_grouping () const
- virtual pattern do_neg_format () const
- virtual string_type do_negative_sign () const
- virtual pattern do_pos_format () const
- virtual string_type do_positive_sign () const
- virtual char_type do_thousands_sep () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.700.1 Detailed Description

template<typename _CharT, bool _Intl>
class std::moneypunct_byname< _CharT, _Intl >

class moneypunct_byname [22.2.6.4].

6.700.2 Member Function Documentation

curr_symbol()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::curr_symbol () const  [inline], [inherited]
```

Return currency symbol string.
This function returns a string_type to use as a currency symbol. It does so by returning returning moneypunct<char_type>::do_curr_symbol().

Returns

string_type representing a currency symbol.

References do_curr_symbol().

decimal_point()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::decimal_point () const  [inline], [inherited]
```

Return decimal point character.
This function returns a char_type to use as a decimal point. It does so by returning returning moneypunct<char_type>::do_decimal_point().

Returns

char_type representing a decimal point.

References do_decimal_point().

do_curr_symbol()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_curr_symbol () const  [inline], [protected],
[virtual], [inherited]
```

Return currency symbol string.
This function returns a string_type to use as a currency symbol. This function is a hook for derived classes to change the value returned.

See also

curr_symbol() for details.

Returns

string_type representing a currency symbol.

Referenced by curr_symbol().

do_decimal_point()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_decimal_point () const  [inline], [protected],
[virtual], [inherited]
```

Return decimal point character.
Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Referenced by decimal_point().

do_frac_digits()

```
template<typename _CharT, bool _Intl>
virtual int std::moneypunct< _CharT, _Intl >::do_frac_digits () const  [inline], [protected],
[virtual], [inherited]
```

Return number of digits in fraction.
This function returns the exact number of digits that make up the fractional part of a money amount. This function is a hook for derived classes to change the value returned.

See also

frac_digits() for details.

Returns

Number of digits in amount fraction.

Referenced by frac_digits().

do_grouping()

```
template<typename _CharT, bool _Intl>
virtual string std::moneypunct< _CharT, _Intl >::do_grouping () const  [inline], [protected],
[virtual], [inherited]
```

Return grouping specification.
Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

grouping() for details.

Returns

String representing grouping specification.

Referenced by grouping().

do_neg_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_neg_format () const  [inline], [protected],
[virtual], [inherited]
```

Return pattern for money values.
This function returns a pattern describing the formatting of a negative valued money amount. This function is a hook for derived classes to change the value returned.

See also

neg_format() for details.

Returns

Pattern for money values.

Referenced by neg_format().

do_negative_sign()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_negative_sign () const  [inline], [protected],
[virtual], [inherited]
```

Return negative sign string.
This function returns a string_type to use as a sign for negative amounts. This function is a hook for derived classes to change the value returned.

See also

negative_sign() for details.

Returns

string_type representing a negative sign.

Referenced by negative_sign().

do_pos_format()

```
template<typename _CharT, bool _Intl>
virtual pattern std::moneypunct< _CharT, _Intl >::do_pos_format () const  [inline], [protected],
[virtual], [inherited]
```

Return pattern for money values.
This function returns a pattern describing the formatting of a positive valued money amount. This function is a hook for derived classes to change the value returned.

See also

pos_format() for details.

Returns

Pattern for money values.

Referenced by pos_format().

do_positive_sign()

```
template<typename _CharT, bool _Intl>
virtual string_type std::moneypunct< _CharT, _Intl >::do_positive_sign () const  [inline], [protected],
[virtual], [inherited]
```

Return positive sign string.
This function returns a string_type to use as a sign for positive amounts. This function is a hook for derived classes to change the value returned.

See also

positive_sign() for details.

Returns

string_type representing a positive sign.

Referenced by positive_sign().

do_thousands_sep()

```
template<typename _CharT, bool _Intl>
virtual char_type std::moneypunct< _CharT, _Intl >::do_thousands_sep () const  [inline], [protected],
[virtual], [inherited]
```

Return thousands separator character.
Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Referenced by thousands_sep().

frac_digits()

```
template<typename _CharT, bool _Intl>
int std::moneypunct< _CharT, _Intl >::frac_digits () const  [inline], [inherited]
```

Return number of digits in fraction.
This function returns the exact number of digits that make up the fractional part of a money amount. It does so by returning returning moneypunct<char_type>::do_frac_digits().
The fractional part of a money amount is optional. But if it is present, there must be frac_digits() digits.

Returns

Number of digits in amount fraction.

References do_frac_digits().

grouping()

```
template<typename _CharT, bool _Intl>
string std::moneypunct< _CharT, _Intl >::grouping () const  [inline], [inherited]
```

Return grouping specification.
This function returns a string representing groupings for the integer part of an amount. Groupings indicate where thousands separators should be inserted.
Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.
For example, if the grouping() returns `\003\002` and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was `32`, this would put more than 50 digits into the least significant group if the character set is ASCII.
The string is returned by calling moneypunct<char_type>::do_grouping().

Returns

string representing grouping specification.

References do_grouping().

neg_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::neg_format () const  [inline], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char_type>::do_pos_format() or moneypunct<char_type>::do_neg_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field. The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr_symbol() may be present. The sign field indicates that the value of positive_sign() or negative_↩ sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos_format() pattern {symbol,sign,value,none}, curr_symbol() == '$' positive_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

References do_neg_format().

negative_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::negative_sign () const  [inline], [inherited]
```

Return negative sign string.

This function returns a string_type to use as a sign for negative amounts. It does so by returning returning moneypunct<char_type>::do_negative_sign().

If the return value contains more than one character, the first character appears in the position indicated by neg_format() and the remainder appear at the end of the formatted string.

Returns

string_type representing a negative sign.

References do_negative_sign().

pos_format()

```
template<typename _CharT, bool _Intl>
pattern std::moneypunct< _CharT, _Intl >::pos_format () const  [inline], [inherited]
```

Return pattern for money values.

This function returns a pattern describing the formatting of a positive or negative valued money amount. It does so by returning returning moneypunct<char_type>::do_pos_format() or moneypunct<char_type>::do_neg_format().

The pattern has 4 fields describing the ordering of symbol, sign, value, and none or space. There must be one of each in the pattern. The none and space enums may not appear in the first field and space may not appear in the final field. The parts of a money string must appear in the order indicated by the fields of the pattern. The symbol field indicates that the value of curr_symbol() may be present. The sign field indicates that the value of positive_sign() or negative_↩ sign() must be present. The value field indicates that the absolute value of the money amount is present. none indicates 0 or more whitespace characters, except at the end, where it permits no whitespace. space indicates that 1 or more whitespace characters must be present.

For example, for the US locale and pos_format() pattern {symbol,sign,value,none}, curr_symbol() == '$' positive_sign() == '+', and value 10.01, and options set to force the symbol, the corresponding string is `$+10.01`.

Returns

Pattern for money values.

References do_pos_format().

positive_sign()

```
template<typename _CharT, bool _Intl>
string_type std::moneypunct< _CharT, _Intl >::positive_sign () const  [inline], [inherited]
```

Return positive sign string.
This function returns a string_type to use as a sign for positive amounts. It does so by returning returning moneypunct<char_type>::do_positive_sign().
If the return value contains more than one character, the first character appears in the position indicated by pos_format() and the remainder appear at the end of the formatted string.

Returns

string_type representing a positive sign.

References do_positive_sign().

thousands_sep()

```
template<typename _CharT, bool _Intl>
char_type std::moneypunct< _CharT, _Intl >::thousands_sep () const  [inline], [inherited]
```

Return thousands separator character.
This function returns a char_type to use as a thousands separator. It does so by returning returning moneypunct<char_type>::do_thousands_sep().

Returns

char_type representing a thousands separator.

References do_thousands_sep().

6.700.3 Member Data Documentation

id

```
template<typename _CharT, bool _Intl>
locale::id std::moneypunct< _CharT, _Intl >::id  [static], [inherited]
```

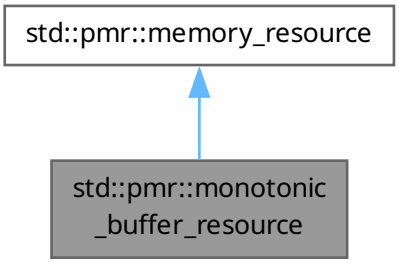
Numpunct facet id.
The documentation for this class was generated from the following file:

- locale_facets_nonio.h

6.701 std::pmr::monotonic_buffer_resource Class Reference

```
#include <memory_resource>
```

Inheritance diagram for std::pmr::monotonic_buffer_resource:



Public Member Functions

- **monotonic_buffer_resource** (const monotonic_buffer_resource &)=delete
- **monotonic_buffer_resource** (memory_resource *__upstream) noexcept
- **monotonic_buffer_resource** (size_t __initial_size) noexcept
- **monotonic_buffer_resource** (size_t __initial_size, memory_resource *__upstream) noexcept
- **monotonic_buffer_resource** (void *__buffer, size_t __buffer_size) noexcept
- **monotonic_buffer_resource** (void *__buffer, size_t __buffer_size, memory_resource *__upstream) noexcept
- void * **allocate** (size_t __bytes, size_t __alignment=_S_max_align)
- void **deallocate** (void *__p, size_t __bytes, size_t __alignment=_S_max_align)
- bool **is_equal** (const memory_resource &__other) const noexcept
- monotonic_buffer_resource & **operator=** (const monotonic_buffer_resource &)=delete
- void **release** () noexcept
- memory_resource * **upstream_resource** () const noexcept

Protected Member Functions

- void * do_allocate (size_t __bytes, size_t __alignment) override
- void do_deallocate (void *, size_t, size_t) override
- bool do_is_equal (const memory_resource &__other) const noexcept override

6.701.1 Detailed Description

A memory resource that allocates from a fixed-size buffer.
The main feature of a `pmr::monotonic_buffer_resource` is that its `do_deallocate` does nothing. This makes it very fast because there is no need to manage a free list, and every allocation simply returns a new block of memory, rather than searching for a suitably-sized free block. Because deallocating is a no-op, the amount of memory used by the resource only grows until `release()` (or the destructor) is called to return all memory to upstream.
A `monotonic_buffer_resource` can be initialized with a buffer that will be used to satisfy all allocation requests, until the buffer is full. After that a new buffer will be allocated from the upstream resource. By using a stack buffer and `pmr::null_memory_resource()` as the upstream you can get a memory resource that only uses the stack and never dynamically allocates.

Since

C++17

6.701.2 Member Function Documentation

do_allocate()

```
void * std::pmr::monotonic_buffer_resource::do_allocate (
            size_t __bytes,
            size_t __alignment)  [inline], [override], [protected], [virtual]
```

Implements std::pmr::memory_resource.

do_deallocate()

```
void std::pmr::monotonic_buffer_resource::do_deallocate (
            void * ,
            size_t ,
            size_t )  [inline], [override], [protected], [virtual]
```

Implements std::pmr::memory_resource.

do_is_equal()

```
bool std::pmr::monotonic_buffer_resource::do_is_equal (
            const memory_resource & __other) const  [inline], [override], [protected], [virtual],
[noexcept]
```

Implements std::pmr::memory_resource.

The documentation for this class was generated from the following file:

- memory_resource

6.702 std::move_iterator< _Iterator > Class Template Reference

`#include <stl_iterator.h>`

Public Types

- using **difference_type**
- using **iterator_concept**
- using **iterator_type**
- using **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- template<typename _Iter>
  constexpr **move_iterator** (const move_iterator< _Iter > &__i)
- constexpr **move_iterator** (iterator_type __i)
- constexpr iterator_type **base** () &&
- constexpr const iterator_type & **base** () const &noexcept
- constexpr reference **operator*** () const
- constexpr move_iterator **operator+** (difference_type __n) const
- constexpr move_iterator & **operator++** ()
- constexpr void **operator++** (int)
- constexpr move_iterator **operator++** (int)
- constexpr move_iterator & **operator+=** (difference_type __n)
- constexpr move_iterator **operator-** (difference_type __n) const
- constexpr move_iterator & **operator--** ()

- constexpr move_iterator **operator--** (int)
- constexpr move_iterator & **operator-=** (difference_type __n)
- constexpr pointer **operator->** () const
- template<typename _Iter>
  constexpr move_iterator & **operator=** (const move_iterator< _Iter > &__i)
- constexpr reference **operator[ ]** (difference_type __n) const

Friends

- constexpr iter_rvalue_reference_t< _Iterator > **iter_move** (const move_iterator &__i) noexcept(noexcept(ranges::↩ iter_move(__i._M_current)))
- template<indirectly_swappable< _Iterator > _Iter2>
  constexpr void **iter_swap** (const move_iterator &__x, const move_iterator< _Iter2 > &__y) noexcept(noexcept(ranges::↩ iter_swap(__x._M_current, __y._M_current)))
- template<sized_sentinel_for< _Iterator > _Sent>
  constexpr iter_difference_t< _Iterator > **operator-** (const move_iterator &__x, const move_sentinel< _Sent > &__y)
- template<sized_sentinel_for< _Iterator > _Sent>
  constexpr iter_difference_t< _Iterator > **operator-** (const move_sentinel< _Sent > &__x, const move_iterator &__y)
- template<sentinel_for< _Iterator > _Sent>
  constexpr bool **operator==** (const move_iterator &__x, const move_sentinel< _Sent > &__y)

6.702.1 Detailed Description

template<typename _Iterator>
class std::move_iterator< _Iterator >

An iterator adaptor that yields an rvalue reference.
Class template move_iterator is an iterator adapter with the same behavior as the underlying iterator except that its dereference operator implicitly converts the value returned by the underlying iterator's dereference operator to an rvalue reference. Some generic algorithms can be called with move iterators to replace copying with moving.
The documentation for this class was generated from the following file:

- bits/stl_iterator.h

6.703 std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> Class Template Reference

`#include <functional>`

Public Types

- using **result_type**

Public Member Functions

- move_only_function () noexcept
- template<typename _Fn, typename _Vt = decay_t<_Fn>>
  move_only_function (_Fn &&__f) noexcept(_S_nothrow_init< _Vt, _Fn >())
- template<typename _Tp, typename... _Args>
  move_only_function (in_place_type_t< _Tp >, _Args &&... __args) noexcept(_S_nothrow_init< _Tp, _Args... >())

- template<typename _Tp, typename _Up, typename... _Args>
  move_only_function (in_place_type_t< _Tp >, initializer_list< _Up > __il, _Args &&... __args) noexcept(_S_↩
  nothrow_init< _Tp, initializer_list< _Up > &, _Args... >())
- move_only_function (move_only_function &&__x) noexcept
- move_only_function (nullptr_t) noexcept
- operator bool () const noexcept
- _Res operator() (_ArgTypes... __args) _GLIBCXX_MOF_CV noexcept(_Noex)
- template<typename _Fn>
  move_only_function & operator= (_Fn &&__f) noexcept(is_nothrow_constructible_v< move_only_function, _Fn >)
- move_only_function & operator= (move_only_function &&__x) noexcept
- move_only_function & operator= (nullptr_t) noexcept
- void swap (move_only_function &__x) noexcept

Friends

- template<typename _Func>
  constexpr auto & **__polyfunc::__base_of** (_Func &) noexcept
- template<typename _Func>
  constexpr auto & **__polyfunc::__invoker_of** (_Func &) noexcept
- template<typename _Src, typename _Dst>
  consteval bool **__polyfunc::__is_invoker_convertible** () noexcept
- bool operator== (const move_only_function &__x, nullptr_t) noexcept
- void swap (move_only_function &__x, move_only_function &__y) noexcept

6.703.1 Detailed Description

template<typename _Res, typename... _ArgTypes, bool _Noex>
class std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

Polymorphic function wrapper.

Since

C++23

The `std::move_only_function` class template is a call wrapper similar to `std::function`, but does not require the stored target function to be copyable.
It also supports const-qualification, ref-qualification, and no-throw guarantees. The qualifications and exception-specification of the `move_only_function::operator()` member function are respected when invoking the target function.

6.703.2 Constructor & Destructor Documentation

move_only_function() [1/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
() [inline], [noexcept]
```

Creates an empty object.
References move_only_function().
Referenced by move_only_function(), move_only_function(), move_only_function(), move_only_function(), move_only_function(), move_only_function(), operator=(), operator=(), operator=(), operator==, swap(), and swap.

move_only_function() [2/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
            nullptr_t ) [inline], [noexcept]
```

Creates an empty object.
References move_only_function().

move_only_function() [3/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
            move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Moves the target object, leaving the source empty.
References move_only_function().

move_only_function() [4/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn, typename _Vt = decay_t<_Fn>>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
            _Fn && __f) [inline], [noexcept]
```

Stores a target object initialized from the argument.
References move_only_function(), and std::forward().

move_only_function() [5/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename... _Args>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
            in_place_type_t< _Tp > ,
            _Args &&... __args) [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.
References move_only_function(), and std::forward().

move_only_function() [6/6]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Tp, typename _Up, typename... _Args>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::move_only_function
(
            in_place_type_t< _Tp > ,
            initializer_list< _Up > __il,
            _Args &&... __args) [inline], [explicit], [noexcept]
```

Stores a target object initialized from the arguments.
References move_only_function(), and std::forward().

6.703.3 Member Function Documentation

operator bool()

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator bool ()
const [inline], [explicit], [noexcept]
```

True if a target object is present, false otherwise.

operator()()

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
_Res std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::operator() (
            _ArgTypes... __args) [inline], [noexcept]
```

Invoke the target object.

The target object will be invoked using the supplied arguments, and as an lvalue or rvalue, and as const or non-const, as dictated by the template arguments of the `move_only_function` specialization.

Precondition

Must not be empty.

References std::forward(), and operator()().

Referenced by operator()().

operator=() [1/3]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
template<typename _Fn>
move_only_function & std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_↩
Noex)>::operator= (
            _Fn && __f) [inline], [noexcept]
```

Stores a new target object, initialized from the argument.

References move_only_function(), std::forward(), and operator=().

operator=() [2/3]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
move_only_function & std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_↩
Noex)>::operator= (
            move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> && __x)
[inline], [noexcept]
```

Stores a new target object, leaving `x` empty.

References move_only_function(), std::addressof(), and operator=().

Referenced by operator=(), operator=(), and operator=().

operator=() [3/3]

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
move_only_function & std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_↩
Noex)>::operator= (
            nullptr_t ) [inline], [noexcept]
```

Destroys the target object (if any).

References move_only_function(), and operator=().

swap()

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>::swap (
            move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x)
[inline], [noexcept]
```

Exchange the target objects (if any).
References move_only_function(), and swap().
Referenced by swap(), and swap.

6.703.4 Friends And Related Symbol Documentation

operator==

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
bool operator== (
            const move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & _↩
_x,
            nullptr_t ) [friend]
```

Check for emptiness by comparing with `nullptr`.
References move_only_function(), and operator==.
Referenced by operator==.

swap

```
template<typename _Res, typename... _ArgTypes, bool _Noex>
void swap (
            move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __x,
            move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)> & __y)
[friend]
```

Exchange the target objects (if any).
References move_only_function(), and swap().
The documentation for this class was generated from the following file:

- mofunc_impl.h

6.704 std::move_sentinel< _Sent > Class Template Reference

`#include <stl_iterator.h>`

Public Member Functions

- constexpr **move_sentinel** (_Sent __s) noexcept(is_nothrow_move_constructible_v< _Sent >)
- template<typename _S2>
  constexpr **move_sentinel** (const move_sentinel< _S2 > &__s) noexcept(is_nothrow_constructible_v< _Sent, const _S2 & >)
- constexpr _Sent **base** () const noexcept(is_nothrow_copy_constructible_v< _Sent >)
- template<typename _S2>
  constexpr move_sentinel & **operator=** (const move_sentinel< _S2 > &__s) noexcept(is_nothrow_assignable_v< _Sent, const _S2 & >)

6.704.1 Detailed Description

template<semiregular _Sent>
class std::move_sentinel< _Sent >

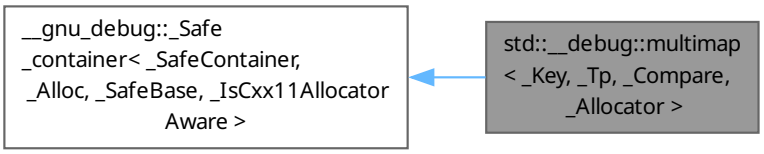
A sentinel adaptor for use with std::move_iterator.

The documentation for this class was generated from the following file:

- bits/stl_iterator.h

6.705 std::__debug::multimap< _Key, _Tp, _Compare, _Allocator > Class Template Reference

`#include <multimap.h>`
Inheritance diagram for std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, multimap > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, multimap > **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef std::pair< const _Key, _Tp > **value_type**

Public Member Functions

- **multimap** (_Base_ref __x)
- template<typename _InputIterator>
  multimap (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator>
  multimap (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- **multimap** (const _Compare &__comp, const _Allocator &__a=_Allocator())
- **multimap** (const allocator_type &__a)
- **multimap** (const multimap &)=default
- **multimap** (const multimap &__m, const __type_identity_t< allocator_type > &__a)

- **multimap** (initializer_list< value_type > __l, const _Compare &__c=_Compare(), const allocator_type &__a=allocator_type())
- **multimap** (initializer_list< value_type > __l, const allocator_type &__a)
- **multimap** (multimap &&)=default
- **multimap** (multimap &&__m, const __type_identity_t< allocator_type > &__a) noexcept(noexcept(_Base(std::move(__m), __a)))
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- _Base_iterator **erase** (_Base_const_iterator __position)
- template<__heterogeneous_tree_key< multimap > _Kt>
  size_type **erase** (_Kt &&__x)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __position)
- iterator **erase** (iterator __position)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (_Pair &&__x)
- iterator **insert** (const value_type &__x)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __position, _Pair &&__x)
- iterator **insert** (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- void **insert** (std::initializer_list< value_type > __list)
- iterator **insert** (value_type &&__x)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- multimap & **operator=** (const multimap &)=default
- multimap & **operator=** (initializer_list< value_type > __l)
- multimap & **operator=** (multimap &&)=default
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rend** () const noexcept

- reverse_iterator **rend** () noexcept
- void **swap** (multimap &__x) noexcept(/∗conditional ∗/)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

6.705.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<std::pair<const _Key, _Tp> >>
class std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >

Class std::multimap with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- multimap.h

6.706 std::multimap< _Key, _Tp, _Compare, _Alloc > Class Template Reference

`#include <map>`

Public Types

- typedef _Alloc **allocator_type**
- typedef _Rep_type::const_iterator **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef _Rep_type::const_reverse_iterator **const_reverse_iterator**
- typedef _Rep_type::difference_type **difference_type**
- typedef _Rep_type::iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Tp **mapped_type**
- typedef _Alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef _Rep_type::reverse_iterator **reverse_iterator**
- typedef _Rep_type::size_type **size_type**
- typedef std::pair< const _Key, _Tp > **value_type**

Public Member Functions

- multimap ()=default
- template<typename _InputIterator>
  multimap (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  multimap (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__↩
  a=allocator_type())

- template<typename _InputIterator>
  multimap (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- multimap (const _Compare &__comp, const allocator_type &__a=allocator_type())
- multimap (const allocator_type &__a)
- multimap (const multimap &)=default
- multimap (const multimap &__m, const __type_identity_t< allocator_type > &__a)
- multimap (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &_↩
  _a=allocator_type())
- multimap (initializer_list< value_type > __l, const allocator_type &__a)
- multimap (multimap &&)=default
- multimap (multimap &&__m, const __type_identity_t< allocator_type > &__a) noexcept(is_nothrow_copy_constructible< _Compare >::value && _Alloc_traits::_S_always_equal())
- ~multimap ()=default
- const_iterator begin () const noexcept
- iterator begin () noexcept
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- const_reverse_iterator crbegin () const noexcept
- const_reverse_iterator crend () const noexcept
- template<typename... _Args>
  iterator emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- const_iterator end () const noexcept
- iterator end () noexcept
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_compare key_comp () const
- size_type max_size () const noexcept
- multimap & operator= (const multimap &)=default
- multimap & operator= (initializer_list< value_type > __l)
- multimap & operator= (multimap &&)=default
- const_reverse_iterator rbegin () const noexcept
- reverse_iterator rbegin () noexcept
- const_reverse_iterator rend () const noexcept
- reverse_iterator rend () noexcept
- size_type size () const noexcept
- void swap (multimap &__x) noexcept(/*conditional */)
- value_compare value_comp () const

- iterator insert (const value_type &__x)
- iterator insert (value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair >::value, iterator > insert (_Pair &&__x)

- iterator insert (const_iterator __position, const value_type &__x)
- iterator insert (const_iterator __position, value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > insert (const_iterator __position, _↩
  Pair &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __position)

- iterator find (const key_type &__x)

- const_iterator find (const key_type &__x) const

- size_type count (const key_type &__x) const

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_t._M_find_tr(__x), void(), true)

- iterator lower_bound (const key_type &__x)

- const_iterator lower_bound (const key_type &__x) const

- iterator upper_bound (const key_type &__x)

- const_iterator upper_bound (const key_type &__x) const

- std::pair< iterator, iterator > equal_range (const key_type &__x)

- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const

Friends

- template<typename _K1, typename _T1, typename _C1, typename _A1>
  bool **operator**< (const multimap< _K1, _T1, _C1, _A1 > &, const multimap< _K1, _T1, _C1, _A1 > &)
- template<typename _K1, typename _T1, typename _C1, typename _A1>
  bool **operator==** (const multimap< _K1, _T1, _C1, _A1 > &, const multimap< _K1, _T1, _C1, _A1 > &)

6.706.1 Detailed Description

template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> > >
class std::multimap< _Key, _Tp, _Compare, _Alloc >

A standard container made up of (key,value) pairs, which can be retrieved based on a key, in logarithmic time.

Since

C++98

Template Parameters

_Key	Type of key objects.
_Tp	Type of mapped objects.
_Compare	Comparison function object type, defaults to less<_Key>.
_Alloc	Allocator type, defaults to allocator<pair<const _Key, _Tp>.

Meets the requirements of a container, a reversible container, and an associative container (using equivalent keys). For a multimap<Key,T> the key_type is Key, the mapped_type is T, and the value_type is std::pair<const Key,T>.
Multimaps support bidirectional iterators.
The private tree data is declared exactly the same way for map and multimap; the distinction is made entirely in how the tree functions are called (*_unique versus *_equal, same as the standard).

6.706.2 Constructor & Destructor Documentation

multimap() [1/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap ()  [default]
```

Default constructor creates no elements.

multimap() [2/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Creates a multimap with no elements.

Parameters

__comp	A comparison object.
__a	An allocator object.

multimap() [3/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            const multimap< _Key, _Tp, _Compare, _Alloc > & )  [default]
```

Multimap copy constructor.
Whether the allocator is copied depends on the allocator traits.

multimap() [4/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            multimap< _Key, _Tp, _Compare, _Alloc > && )  [default]
```

Multimap move constructor.
The newly-created multimap contains the exact contents of the moved instance. The moved instance is a valid, but unspecified multimap.

multimap() [5/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            initializer_list< value_type > __l,
            const _Compare & __comp = _Compare(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a multimap from an initializer_list.

Parameters

__l	An initializer_list.
__comp	A comparison functor.
__a	An allocator object.

Create a multimap consisting of copies of the elements from the initializer_list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is *__l.size()*).

multimap() [6/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            const allocator_type & __a)  [inline], [explicit]
```

Allocator-extended default constructor.

multimap() [7/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            const multimap< _Key, _Tp, _Compare, _Alloc > & __m,
            const __type_identity_t< allocator_type > & __a)  [inline]
```

Allocator-extended copy constructor.

multimap() [8/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            multimap< _Key, _Tp, _Compare, _Alloc > && __m,
            const __type_identity_t< allocator_type > & __a)  [inline], [noexcept]
```

Allocator-extended move constructor.

multimap() [9/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            initializer_list< value_type > __l,
            const allocator_type & __a)  [inline]
```

Allocator-extended initializer-list constructor.

multimap() [10/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a)  [inline]
```

Allocator-extended range constructor.

multimap() [11/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Builds a multimap from a range.

Parameters

__first	An input iterator.
__last	An input iterator.

Create a multimap consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

multimap() [12/12]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
```

```
std::multimap< _Key, _Tp, _Compare, _Alloc >::multimap (
            _InputIterator __first,
            _InputIterator __last,
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a multimap from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__comp	A comparison functor.
__a	An allocator object.

Create a multimap consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

~multimap()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::multimap< _Key, _Tp, _Compare, _Alloc >::~multimap ()  [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

6.706.3 Member Function Documentation

begin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

begin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::begin ()  [inline], [noexcept]
```

Returns a read/write iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

cbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first pair in the multimap. Iteration is done in ascending order according to the keys.

cend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

clear()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Kt>
auto std::multimap< _Key, _Tp, _Compare, _Alloc >::contains (
             const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void(), true)  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

True if there is any element with the specified key.

contains() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::multimap< _Key, _Tp, _Compare, _Alloc >::contains (
             const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

True if there is any element with the specified key.

count()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements with given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Number of elements with specified key.

crbegin()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

crend()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::crend () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

emplace()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::emplace (
            _Args &&... __args)  [inline]
```

Build and insert a std::pair into the multimap.

Parameters

__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the inserted (key,value) pair.

This function builds and inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.
Insertion requires logarithmic time.

emplace_hint()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename... _Args>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&... __args)  [inline]
```

Builds and inserts a std::pair into the multimap.

Parameters

__pos	An iterator that serves as a hint as to where the pair should be inserted.
__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↩
html#containers.associative.insert_hints
Insertion requires logarithmic time (if the hint is not taken).

empty()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
bool std::multimap< _Key, _Tp, _Compare, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the multimap is empty.

end() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

end() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::end ()  [inline], [noexcept]
```

Returns a read/write iterator that points one past the last pair in the multimap. Iteration is done in ascending order according to the keys.

equal_range() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< iterator, iterator > std::multimap< _Key, _Tp, _Compare, _Alloc >::equal_range (
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

equal_range() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
std::pair< const_iterator, const_iterator > std::multimap< _Key, _Tp, _Compare, _Alloc >::↩
equal_range (
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pairs to be located.

Returns

Pair of read-only (constant) iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).

erase() [1/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _*x*	Key of element to be erased.

Returns

The number of elements erased.

This function erases all elements located by the given key from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [first,last) range of elements from a multimap.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased .

Returns

The iterator *__last*.

This function erases a sequence of elements from a multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from a multimap.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [4/4]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::erase (
            iterator __position)  [inline]
```

Erases an element from a multimap.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in a multimap.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
```

```
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in a multimap.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns a constant iterator pointing to the sought after pair. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
allocator_type std::multimap< _Key, _Tp, _Compare, _Alloc >::get_allocator () const  [inline],
[noexcept]
```

Get a copy of the memory allocation object.

insert() [1/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _InputIterator>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair >::value, iterator > std::multimap< _Key, _↩
Tp, _Compare, _Alloc >::insert (
            _Pair && __x)  [inline]
```

Inserts a std::pair into the multimap.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.
Insertion requires logarithmic time.

insert() [3/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Inserts a std::pair into the multimap.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.
Insertion requires logarithmic time.
Referenced by std::multimap< _Key, _Tp, _Cmp, polymorphic_allocator< pair< const _Key, _Tp > > >::insert().

insert() [4/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::multimap< _Key,
_Tp, _Compare, _Alloc >::insert (
            const_iterator __position,
            _Pair && __x)  [inline]
```

Inserts a std::pair into the multimap.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires logarithmic time (if the hint is not taken).

insert() [5/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
            const_iterator __position,
            const value_type & __x)  [inline]
```

Inserts a std::pair into the multimap.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires logarithmic time (if the hint is not taken).

insert() [6/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
            const_iterator __position,
            value_type && __x)  [inline]
```

Inserts a std::pair into the multimap.

Parameters

__position	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires logarithmic time (if the hint is not taken).

insert() [7/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Attempts to insert a list of std::pairs into the multimap.

Parameters

↩ _↩ ↩ _↩ *l*	A std::initializer_list<value_type> of pairs to be inserted.

Complexity similar to that of the range constructor.

insert() [8/8]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::insert (
            value_type && __x)  [inline]
```

Inserts a std::pair into the multimap.

Parameters

_↩ *_x*	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted (key,value) pair.

This function inserts a (key, value) pair into the multimap. Contrary to a std::map the multimap does not rely on unique keys and thus multiple pairs with the same key can be inserted.
Insertion requires logarithmic time.

key_comp()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
key_compare std::multimap< _Key, _Tp, _Compare, _Alloc >::key_comp () const  [inline]
```

Returns the key comparison object out of which the multimap was constructed.

lower_bound() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
            const key_type & __x)  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

lower_bound() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::lower_bound (
            const key_type & __x) const  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful the iterator will point to the next greatest element or, if no such greater element exists, to end().

max_size()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the maximum size of the multimap.

operator=() [1/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap & std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (
            const multimap< _Key, _Tp, _Compare, _Alloc > & )  [default]
```

Multimap assignment operator.
Whether the allocator is copied depends on the allocator traits.

operator=() [2/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap & std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Multimap list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a multimap with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the multimap and that the resulting multimap's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
multimap & std::multimap< _Key, _Tp, _Compare, _Alloc >::operator= (
            multimap< _Key, _Tp, _Compare, _Alloc > && )  [default]
```

Move assignment operator.

rbegin() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

rbegin() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rbegin ()  [inline], [noexcept]
```

Returns a read/write reverse iterator that points to the last pair in the multimap. Iteration is done in descending order according to the keys.

rend() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend () const  [inline],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

rend() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
reverse_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::rend ()  [inline], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first pair in the multimap. Iteration is done in descending order according to the keys.

size()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
size_type std::multimap< _Key, _Tp, _Compare, _Alloc >::size () const  [inline], [noexcept]
```

Returns the size of the multimap.

swap()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
void std::multimap< _Key, _Tp, _Compare, _Alloc >::swap (
            multimap< _Key, _Tp, _Compare, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another multimap.

Parameters

_↩ _x	A multimap of the same element and allocator types.

This exchanges the elements between two multimaps in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.
Referenced by std::swap().

upper_bound() [1/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
            const key_type & __x)  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [2/2]

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
const_iterator std::multimap< _Key, _Tp, _Compare, _Alloc >::upper_bound (
            const key_type & __x) const  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key of (key, value) pair to be located.

Returns

Read-only (constant) iterator pointing to first iterator greater than key, or end().

value_comp()

```
template<typename _Key, typename _Tp, typename _Compare = std::less<_Key>, typename _Alloc =
std::allocator<std::pair<const _Key, _Tp> >>
value_compare std::multimap< _Key, _Tp, _Compare, _Alloc >::value_comp () const  [inline]
```

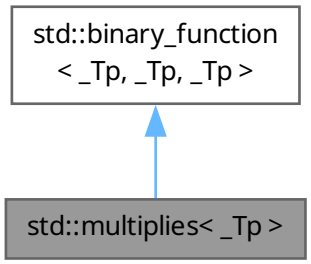
Returns a value comparison object, built from the key comparison object out of which the multimap was constructed.
The documentation for this class was generated from the following files:

- stl_map.h
- stl_multimap.h

6.707 std::multiplies< _Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::multiplies< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr _Tp **operator()** (const _Tp &__x, const _Tp &__y) const

6.707.1 Detailed Description

template<typename _Tp>
struct std::multiplies< _Tp >

One of the math functors.

6.707.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

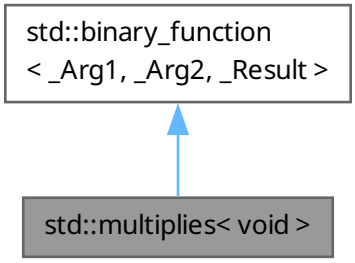
```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.708 std::multiplies< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::multiplies< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**

- typedef void result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) *std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) *std::forward< _Up >(__u))

6.708.1 Detailed Description

One of the math functors.

6.708.2 Member Typedef Documentation

first_argument_type

`typedef void std::binary_function< void, void, void >::first_argument_type`
`first_argument_type` is the type of the first argument

result_type

`typedef void std::binary_function< void, void, void >::result_type`
`result_type` is the return type

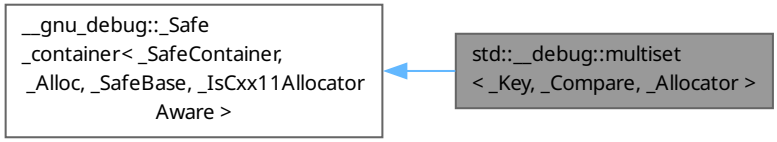
second_argument_type

`typedef void std::binary_function< void, void, void >::second_argument_type`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.709 std::__debug::multiset< _Key, _Compare, _Allocator > Class Template Reference

`#include <multiset.h>`
Inheritance diagram for std::__debug::multiset< _Key, _Compare, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, multiset > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**

- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, multiset > **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **multiset** (_Base_ref __x)
- template<typename _InputIterator>
  multiset (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator>
  multiset (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- **multiset** (const _Compare &__comp, const _Allocator &__a=_Allocator())
- **multiset** (const allocator_type &__a)
- **multiset** (const multiset &)=default
- **multiset** (const multiset &__m, const __type_identity_t< allocator_type > &__a)
- **multiset** (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- **multiset** (initializer_list< value_type > __l, const allocator_type &__a)
- **multiset** (multiset &&)=default
- **multiset** (multiset &&__m, const __type_identity_t< allocator_type > &__a) noexcept(noexcept(_Base(std::move(__m), __a)))
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- _Base_iterator **erase** (_Base_const_iterator __position)
- template<__heterogeneous_tree_key< multiset > _Kt>
  size_type **erase** (_Kt &&__x)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)

- iterator **erase** (const_iterator __position)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- iterator **insert** (const value_type &__x)
- iterator **insert** (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- void **insert** (initializer_list< value_type > __l)
- iterator **insert** (value_type &&__x)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- multiset & **operator=** (const multiset &)=default
- multiset & **operator=** (initializer_list< value_type > __l)
- multiset & **operator=** (multiset &&)=default
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rend** () const noexcept
- reverse_iterator **rend** () noexcept
- void **swap** (multiset &__x) noexcept(/*conditional */)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

6.709.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>
class std::__debug::multiset< _Key, _Compare, _Allocator >

Class std::multiset with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- multiset.h

6.710 std::multiset< _Key, _Compare, _Alloc > Class Template Reference

`#include <set>`

Public Types

- typedef _Alloc **allocator_type**
- typedef _Rep_type::const_iterator **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef _Rep_type::const_reverse_iterator **const_reverse_iterator**

- typedef _Rep_type::difference_type **difference_type**
- typedef _Rep_type::const_iterator **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Alloc_traits::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef _Rep_type::const_reverse_iterator **reverse_iterator**
- typedef _Rep_type::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- multiset ()=default
- template<typename _InputIterator>
  multiset (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  multiset (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__↩
  a=allocator_type())
- template<typename _InputIterator>
  multiset (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- multiset (const _Compare &__comp, const allocator_type &__a=allocator_type())
- multiset (const allocator_type &__a)
- multiset (const multiset &)=default
- multiset (const multiset &__m, const __type_identity_t< allocator_type > &__a)
- multiset (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__↩
  a=allocator_type())
- multiset (initializer_list< value_type > __l, const allocator_type &__a)
- multiset (multiset &&)=default
- multiset (multiset &&__m, const __type_identity_t< allocator_type > &__a) noexcept(is_nothrow_copy_constructible< _Compare >::value &&_Alloc_traits::_S_always_equal())
- ~multiset ()=default
- iterator begin () const noexcept
- iterator cbegin () const noexcept
- iterator cend () const noexcept
- void clear () noexcept
- reverse_iterator crbegin () const noexcept
- reverse_iterator crend () const noexcept
- template<typename... _Args>
  iterator emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- iterator end () const noexcept
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- iterator erase (const_iterator __position)
- allocator_type get_allocator () const noexcept
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- iterator insert (const value_type &__x)
- iterator insert (const_iterator __position, const value_type &__x)

- iterator **insert** (const_iterator __position, value_type &&__x)
- void insert (initializer_list< value_type > __l)
- iterator **insert** (value_type &&__x)
- key_compare key_comp () const
- size_type max_size () const noexcept
- multiset & operator= (const multiset &)=default
- multiset & operator= (initializer_list< value_type > __l)
- multiset & operator= (multiset &&)=default
- reverse_iterator rbegin () const noexcept
- reverse_iterator rend () const noexcept
- size_type size () const noexcept
- void swap (multiset &__x) noexcept(/*conditional */)
- value_compare value_comp () const

- size_type count (const key_type &__x) const

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_t._M_find_tr(__x), void(), true)

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const

- iterator lower_bound (const key_type &__x)
- const_iterator lower_bound (const key_type &__x) const

- iterator upper_bound (const key_type &__x)
- const_iterator upper_bound (const key_type &__x) const

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const

Friends

- template<typename _K1, typename _C1, typename _A1>
  bool **operator**< (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)
- template<typename _K1, typename _C1, typename _A1>
  bool **operator==** (const multiset< _K1, _C1, _A1 > &, const multiset< _K1, _C1, _A1 > &)

6.710.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
class std::multiset< _Key, _Compare, _Alloc >

A standard container made up of elements, which can be retrieved in logarithmic time.

Since

C++98

Template Parameters

_Key	Type of key objects.
_Compare	Comparison function object type, defaults to less<_Key>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a container, a reversible container, and an associative container (using equivalent keys). For a `multiset<Key>` the key_type and value_type are Key.
Multisets support bidirectional iterators.
The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (*_unique versus *_equal, same as the standard).

6.710.2 Constructor & Destructor Documentation

multiset() [1/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset ()  [default]
```

Default constructor creates no elements.

multiset() [2/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Creates a multiset with no elements.

Parameters

__comp	Comparator to use.
__a	An allocator object.

multiset() [3/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
std::multiset< _Key, _Compare, _Alloc >::multiset (
```

```
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Builds a multiset from a range.

Parameters

__first	An input iterator.
__last	An input iterator.

Create a multiset consisting of copies of the elements from [first,last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

multiset() [4/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            _InputIterator __first,
            _InputIterator __last,
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a multiset from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__comp	A comparison functor.
__a	An allocator object.

Create a multiset consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

multiset() [5/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            const multiset< _Key, _Compare, _Alloc > & )  [default]
```

Multiset copy constructor.
Whether the allocator is copied depends on the allocator traits.

multiset() [6/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            multiset< _Key, _Compare, _Alloc > && )  [default]
```

Multiset move constructor.

The newly-created multiset contains the exact contents of the moved instance. The moved instance is a valid, but unspecified multiset.

multiset() [7/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            initializer_list< value_type > __l,
            const _Compare & __comp = _Compare(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a multiset from an initializer_list.

Parameters

__l	An initializer_list.
__comp	A comparison functor.
__a	An allocator object.

Create a multiset consisting of copies of the elements from the list. This is linear in N if the list is already sorted, and NlogN otherwise (where N is *__l.size()*).

multiset() [8/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            const allocator_type & __a)  [inline], [explicit]
```

Allocator-extended default constructor.

multiset() [9/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            const multiset< _Key, _Compare, _Alloc > & __m,
            const __type_identity_t< allocator_type > & __a)  [inline]
```

Allocator-extended copy constructor.

multiset() [10/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            multiset< _Key, _Compare, _Alloc > && __m,
            const __type_identity_t< allocator_type > & __a)  [inline], [noexcept]
```

Allocator-extended move constructor.

multiset() [11/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            initializer_list< value_type > __l,
            const allocator_type & __a)  [inline]
```

Allocator-extended initializer-list constructor.

multiset() [12/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
std::multiset< _Key, _Compare, _Alloc >::multiset (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a)  [inline]
```

Allocator-extended range constructor.

~multiset()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::multiset< _Key, _Compare, _Alloc >::~multiset ()  [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

6.710.3 Member Function Documentation

begin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::begin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

cbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the multiset. Iteration is done in ascending order according to the keys.

cend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

clear()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
void std::multiset< _Key, _Compare, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::multiset< _Key, _Compare, _Alloc >::contains (
            const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void(), true)  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

contains() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
bool std::multiset< _Key, _Compare, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

count()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements with given key.

Parameters

_↩ _x	Key of elements to be located.

Returns

Number of elements with specified key.

crbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::crbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

crend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::crend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

emplace()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename... _Args>
iterator std::multiset< _Key, _Compare, _Alloc >::emplace (
            _Args &&... __args)  [inline]
```

Builds and inserts an element into the multiset.

Parameters

__args	Arguments used to generate the element instance to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.
Insertion requires logarithmic time.

emplace_hint()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename... _Args>
iterator std::multiset< _Key, _Compare, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&... __args)  [inline]
```

Builds and inserts an element into the multiset.

Parameters

__pos	An iterator that serves as a hint as to where the element should be inserted.
__args	Arguments used to generate the element instance to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.
Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires logarithmic time (if the hint is not taken).

empty()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
bool std::multiset< _Key, _Compare, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the set is empty.

end()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the multiset. Iteration is done in ascending order according to the keys.

equal_range() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
std::pair< iterator, iterator > std::multiset< _Key, _Compare, _Alloc >::equal_range (
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

__x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multisets.

equal_range() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::pair< const_iterator, const_iterator > std::multiset< _Key, _Compare, _Alloc >::equal_range
(
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multisets.

erase() [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all elements located by the given key from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [first,last) range of elements from a multiset.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *last*.

This function erases a sequence of elements from a multiset. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from a multiset.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a multiset. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in a set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in a set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
allocator_type std::multiset< _Key, _Compare, _Alloc >::get_allocator () const  [inline], [noexcept]
```

Returns the memory allocation object.

insert() [1/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
void std::multiset< _Key, _Compare, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that tries to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Inserts an element into the multiset.

Parameters

_↩ _x	Element to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.
Insertion requires logarithmic time.
Referenced by std::multiset< _Key, _Cmp, polymorphic_allocator< _Key > >::insert().

insert() [3/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::insert (
            const_iterator __position,
            const value_type & __x)  [inline]
```

Inserts an element into the multiset.

Parameters

__position	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the inserted element.

This function inserts an element into the multiset. Contrary to a std::set the multiset does not rely on unique keys and thus multiple copies of the same element can be inserted.
Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↩associative.insert_hints for more on *hinting*.
Insertion requires logarithmic time (if the hint is not taken).

insert() [4/4]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
void std::multiset< _Key, _Compare, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Attempts to insert a list of elements into the multiset.

Parameters

↩ _↩ ↩ _↩ *l*	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

key_comp()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
key_compare std::multiset< _Key, _Compare, _Alloc >::key_comp () const  [inline]
```

Returns the comparison object.

lower_bound() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::lower_bound (
            const key_type & __x)  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ *_x*	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

lower_bound() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::lower_bound (
            const key_type & __x) const  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ *_x*	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

max_size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the maximum size of the set.

operator=() [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
multiset & std::multiset< _Key, _Compare, _Alloc >::operator= (
            const multiset< _Key, _Compare, _Alloc > & )  [default]
```

Multiset assignment operator.
Whether the allocator is copied depends on the allocator traits.

operator=() [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
multiset & std::multiset< _Key, _Compare, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Multiset list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a multiset with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the multiset and that the resulting multiset's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
multiset & std::multiset< _Key, _Compare, _Alloc >::operator= (
            multiset< _Key, _Compare, _Alloc > && )  [default]
```

Move assignment operator.

rbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
```

```
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rbegin () const  [inline], [noexcept]
```
Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

rend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::multiset< _Key, _Compare, _Alloc >::rend () const  [inline], [noexcept]
```
Returns a read-only (constant) reverse iterator that points to the last element in the multiset. Iteration is done in descending order according to the keys.

size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::multiset< _Key, _Compare, _Alloc >::size () const  [inline], [noexcept]
```
Returns the size of the set.

swap()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
void std::multiset< _Key, _Compare, _Alloc >::swap (
            multiset< _Key, _Compare, _Alloc > & __x)  [inline], [noexcept]
```
Swaps data with another multiset.

Parameters

_↩ _x	A multiset of the same element and allocator types.

This exchanges the elements between two multisets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.
Referenced by std::swap().

upper_bound() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (
            const key_type & __x)  [inline]
```
Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
const_iterator std::multiset< _Key, _Compare, _Alloc >::upper_bound (
            const key_type & __x) const  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

value_comp()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
value_compare std::multiset< _Key, _Compare, _Alloc >::value_comp () const  [inline]
```

Returns the comparison object.

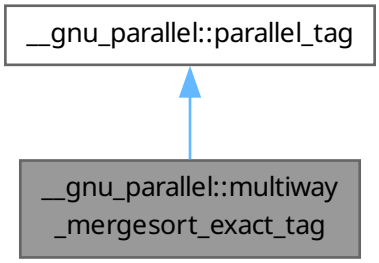
The documentation for this class was generated from the following file:

- stl_multiset.h

6.711 __gnu_parallel::multiway_mergesort_exact_tag Struct Reference

`#include <tags.h>`

Inheritance diagram for __gnu_parallel::multiway_mergesort_exact_tag:



Public Member Functions

- **multiway_mergesort_exact_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

6.711.1 Detailed Description

Forces parallel sorting using multiway mergesort with exact splitting at compile time.

6.711.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

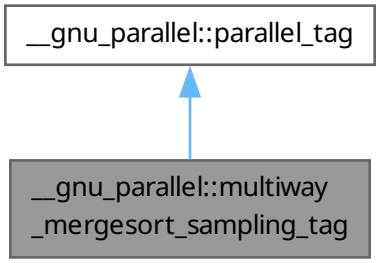
The documentation for this struct was generated from the following file:

- tags.h

6.712 __gnu_parallel::multiway_mergesort_sampling_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::multiway_mergesort_sampling_tag:



Public Member Functions

- **multiway_mergesort_sampling_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

6.712.1 Detailed Description

Forces parallel sorting using multiway mergesort with splitting by sampling at compile time.

6.712.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

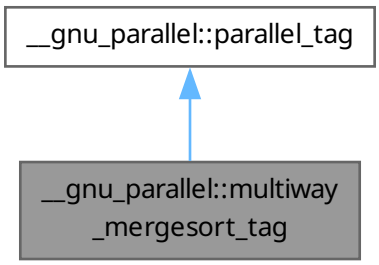
__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

6.713 __gnu_parallel::multiway_mergesort_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::multiway_mergesort_tag:



Public Member Functions

- **multiway_mergesort_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

6.713.1 Detailed Description

Forces parallel sorting using multiway mergesort at compile time.

6.713.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

6.714 std::mutex Class Reference

`#include <mutex>`

Public Types

- typedef __native_type * **native_handle_type**

Public Member Functions

- **mutex** (const mutex &)=delete
- void **lock** ()
- native_handle_type **native_handle** () noexcept
- mutex & **operator=** (const mutex &)=delete
- bool **try_lock** () noexcept
- void **unlock** ()

6.714.1 Detailed Description

The standard mutex type.
A simple, non-recursive, non-timed mutex.
Do not call `lock()` and `unlock()` directly, use a scoped lock type such as `std::unique_lock`, `std::lock_guard`, or (since C++17) `std::scoped_lock`.

Since

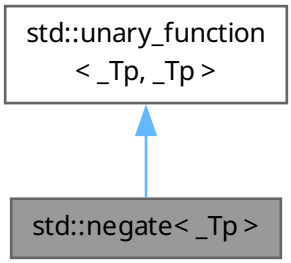
C++11

The documentation for this class was generated from the following file:

- std_mutex.h

6.715 std::negate< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::negate< _Tp >:



Public Types

- typedef _Tp argument_type
- typedef _Tp result_type

Public Member Functions

- constexpr _Tp **operator()** (const _Tp &__x) const

6.715.1 Detailed Description

template<**typename _Tp**>
struct std::negate< **_Tp** >

One of the math functors.

6.715.2 Member Typedef Documentation

argument_type

```
typedef _Tp std::unary_function< _Tp, _Tp >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

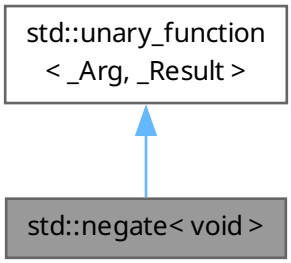
```
typedef _Tp std::unary_function< _Tp, _Tp >::result_type  [inherited]
```

`result_type` is the return type
The documentation for this struct was generated from the following file:

- stl_function.h

6.716 std::negate< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::negate< void >:



Public Types

- typedef void argument_type
- typedef __is_transparent **is_transparent**
- typedef void result_type

Public Member Functions

- template<typename _Tp>
  constexpr auto **operator()** (_Tp &&__t) const noexcept(noexcept(-std::forward< _Tp >(__t))) -> decltype(-std::forward< _Tp >(__t))

6.716.1 Detailed Description

One of the math functors.

6.716.2 Member Typedef Documentation

argument_type

```
typedef void std::unary_function< void, void >::argument_type
```

`argument_type` is the type of the argument

result_type

```
typedef void std::unary_function< void, void >::result_type
```

`result_type` is the return type

The documentation for this struct was generated from the following file:

- stl_function.h

6.717 std::negative_binomial_distribution< _IntType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- **negative_binomial_distribution** (_IntType __k, double __p=0.5)
- **negative_binomial_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _IntType k () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)

- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- double p () const
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::negative_binomial_distribution< _IntType1 > &__x)
- bool operator== (const negative_binomial_distribution &__d1, const negative_binomial_distribution &__d2)
- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::negative_binomial_distribution< _IntType1 > &__x)

6.717.1 Detailed Description

template<typename _IntType = int>
class std::negative_binomial_distribution< _IntType >

A negative_binomial_distribution random number distribution.
The formula for the negative binomial probability mass function is $p(i) = \binom{n}{i} p^i (1-p)^{t-i}$ where t and p are the parameters of the distribution.

Since

C++11

6.717.2 Member Typedef Documentation

result_type

```
template<typename _IntType = int>
typedef _IntType std::negative_binomial_distribution< _IntType >::result_type
```

The type of the range of the distribution.

6.717.3 Member Function Documentation

k()

```
template<typename _IntType = int>
_IntType std::negative_binomial_distribution< _IntType >::k () const  [inline]
```

Return the k parameter of the distribution.

max()

```
template<typename _IntType = int>
result_type std::negative_binomial_distribution< _IntType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _IntType = int>
result_type std::negative_binomial_distribution< _IntType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _IntType>
template<typename _UniformRandomNumberGenerator>
negative_binomial_distribution< _IntType >::result_type std::negative_binomial_distribution< _↩
IntType >::operator() (
            _UniformRandomNumberGenerator & __urng)
```

Generating functions.

p()

```
template<typename _IntType = int>
double std::negative_binomial_distribution< _IntType >::p () const  [inline]
```

Return the p parameter of the distribution.

param() [1/2]

```
template<typename _IntType = int>
param_type std::negative_binomial_distribution< _IntType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _IntType = int>
void std::negative_binomial_distribution< _IntType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _IntType = int>
void std::negative_binomial_distribution< _IntType >::reset ()  [inline]
```

Resets the distribution state.

6.717.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::negative_binomial_distribution< _IntType1 > & __x)  [friend]
```

Inserts a negative_binomial_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.

__x	A negative_binomial_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _IntType = int>
bool operator== (
             const negative_binomial_distribution< _IntType > & __d1,
             const negative_binomial_distribution< _IntType > & __d2)  [friend]
```

Return true if two negative binomial distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
             std::basic_istream< _CharT, _Traits > & __is,
             std::negative_binomial_distribution< _IntType1 > & __x)  [friend]
```

Extracts a negative_binomial_distribution random number distribution __x from the input stream __is.

Parameters

_is	An input stream.
_x	A negative_binomial_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.718 std::nested_exception Class Reference

```
#include <exception>
```

Public Member Functions

- nested_exception () noexcept
- **nested_exception** (const nested_exception &) noexcept=default
- exception_ptr nested_ptr () const noexcept
- nested_exception & **operator=** (const nested_exception &) noexcept=default
- void rethrow_nested () const

6.718.1 Detailed Description

Mixin class that stores the current exception.
This type can be used via `std::throw_with_nested` to store the current exception nested within another exception.

Since

C++11

See also

std::throw_with_nested

6.718.2 Constructor & Destructor Documentation

nested_exception()

`std::nested_exception::nested_exception () [inline], [noexcept]`
The default constructor stores the current exception (if any).
References std::current_exception().

6.718.3 Member Function Documentation

nested_ptr()

`exception_ptr std::nested_exception::nested_ptr () const [inline], [noexcept]`
Access the stored exception.

rethrow_nested()

`void std::nested_exception::rethrow_nested () const [inline]`
Rethrow the stored exception, or terminate if none was stored.
References std::rethrow_exception(), rethrow_nested(), and std::terminate().
Referenced by rethrow_nested().
The documentation for this class was generated from the following file:

- nested_exception.h

6.719 __gnu_cxx::limit_condition::never_adjustor Struct Reference

`#include <throw_allocator.h>`

6.719.1 Detailed Description

Never enter the condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.720 __gnu_cxx::random_condition::never_adjustor Struct Reference

`#include <throw_allocator.h>`

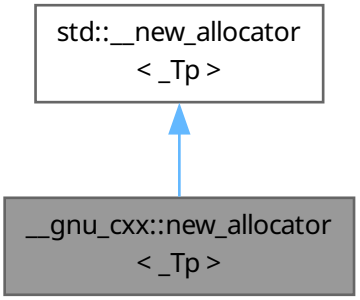
6.720.1 Detailed Description

Never enter the condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.721 __gnu_cxx::new_allocator< _Tp > Class Template Reference

`#include <ext/new_allocator.h>`
Inheritance diagram for __gnu_cxx::new_allocator< _Tp >:



Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **new_allocator** (const new_allocator &) noexcept
- template<typename _Tp1>
  new_allocator (const new_allocator< _Tp1 > &) noexcept
- constexpr _Tp * **allocate** (size_type __n, const void *=static_cast< const void * >(0))
- constexpr std::allocation_result< _Tp *, size_t > **allocate_at_least** (size_t __n)
- constexpr void **deallocate** (_Tp *__p, size_type __n)

Static Public Member Functions

- static constexpr void **_S_check_allocation_limit** (size_t __n)

6.721.1 Detailed Description

template<typename _Tp>
class __gnu_cxx::new_allocator< _Tp >

An allocator that uses global `new`, as per C++03 [20.4.1].
This is precisely the allocator defined in the C++ Standard.

- all allocation calls `operator new`
- all deallocation calls `operator delete`

This is a non-standard extension that can be used to guarantee allocation from `new` even if the library has been configured to use a different implementation for `std::allocator`.

Template Parameters

_Tp	Type of allocated object.

The documentation for this class was generated from the following file:

- ext/new_allocator.h

6.722 __gnu_pbds::detail::no_throw_copies< Key, Mapped > Struct Template Reference

`#include <types_traits.hpp>`

Public Types

- typedef integral_constant< int, __simple > **indicator**

Static Public Attributes

- static const bool **__simple**

6.722.1 Detailed Description

template<typename Key, typename Mapped>
struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >

Primary template.
The documentation for this struct was generated from the following file:

- types_traits.hpp

6.723 __gnu_pbds::detail::no_throw_copies< Key, null_type > Struct Template Reference

`#include <types_traits.hpp>`

Public Types

- typedef integral_constant< int, is_simple< Key >::value > **indicator**

Static Public Attributes

- static const bool **__simple**

6.723.1 Detailed Description

template<typename Key>
struct __gnu_pbds::detail::no_throw_copies< Key, null_type >

Specialization.
The documentation for this struct was generated from the following file:

- types_traits.hpp

6.724 std::normal_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **normal_distribution** (const param_type &__p)
- normal_distribution (result_type __mean, result_type __stddev=result_type(1))
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- _RealType mean () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()
- _RealType stddev () const

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::normal_distribution< _RealType1 > &__x)
- template<typename _RealType1>
  bool operator== (const std::normal_distribution< _RealType1 > &__d1, const std::normal_distribution< _Real↩ Type1 > &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::normal_distribution< _RealType1 > &__x)

6.724.1 Detailed Description

template<typename _RealType = double>
class std::normal_distribution< _RealType >

A normal continuous distribution for random numbers.
The formula for the normal probability density function is

$$p(x|\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x-\mu^2}{2\sigma^2}}$$

Since

C++11

6.724.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::normal_distribution< _RealType >::result_type
```

The type of the range of the distribution.

6.724.3 Constructor & Destructor Documentation

normal_distribution()

```
template<typename _RealType = double>
std::normal_distribution< _RealType >::normal_distribution (
            result_type __mean,
            result_type __stddev = result_type(1))  [inline], [explicit]
```

Constructs a normal distribution with parameters $mean$ and standard deviation.

6.724.4 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::normal_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

mean()

```
template<typename _RealType = double>
_RealType std::normal_distribution< _RealType >::mean () const  [inline]
```

Returns the mean of the distribution.

min()

```
template<typename _RealType = double>
result_type std::normal_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()() [1/2]

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::normal_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

operator()() [2/2]

```
template<typename _RealType>
template<typename _UniformRandomNumberGenerator>
normal_distribution< _RealType >::result_type std::normal_distribution< _RealType >::operator()
```

```
(
            _UniformRandomNumberGenerator & __urng,
            const param_type & __param)
```

Polar method due to Marsaglia.
Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. V, Sect. 4.4.
References std::log(), and std::sqrt().

param() [1/2]

```
template<typename _RealType = double>
param_type std::normal_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::normal_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::normal_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

stddev()

```
template<typename _RealType = double>
_RealType std::normal_distribution< _RealType >::stddev () const  [inline]
```

Returns the standard deviation of the distribution.

6.724.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::normal_distribution< _RealType1 > & __x)  [friend]
```

Inserts a normal_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A normal_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
template<typename _RealType1>
bool operator== (
             const std::normal_distribution< _RealType1 > & __d1,
             const std::normal_distribution< _RealType1 > & __d2)  [friend]
```

Return true if two normal distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
             std::basic_istream< _CharT, _Traits > & __is,
             std::normal_distribution< _RealType1 > & __x)  [friend]
```

Extracts a normal_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A normal_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

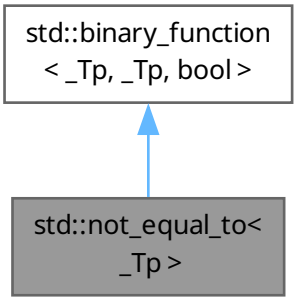
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.725 std::not_equal_to< _Tp > Struct Template Reference

```
#include <stl_function.h>
```

Inheritance diagram for std::not_equal_to< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr bool **operator()** (const _Tp &__x, const _Tp &__y) const

6.725.1 Detailed Description

template<typename _Tp>
struct std::not_equal_to< _Tp >

One of the comparison functors.

6.725.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.726 std::ranges::not_equal_to Struct Reference

`#include <ranges_cmp.h>`

Public Types

- using **is_transparent**

Static Public Member Functions

- template<typename _Tp, typename _Up>
  static constexpr bool **operator()** (_Tp &&__t, _Up &&__u) noexcept(noexcept(std::declval< _Tp >()==std::declval< _Up >()))

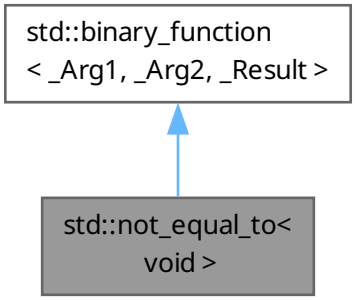
6.726.1 Detailed Description

ranges::not_equal_to function object type.
The documentation for this struct was generated from the following file:

- ranges_cmp.h

6.727 std::not_equal_to< void > Struct Reference

`#include <stl_function.h>`
Inheritance diagram for std::not_equal_to< void >:



Public Types

- typedef void first_argument_type
- typedef __is_transparent **is_transparent**
- typedef bool result_type
- typedef void second_argument_type

Public Member Functions

- template<typename _Tp, typename _Up>
  constexpr auto **operator()** (_Tp &&__t, _Up &&__u) const noexcept(noexcept(std::forward< _Tp >(__t) !=std::forward< _Up >(__u))) -> decltype(std::forward< _Tp >(__t) !=std::forward< _Up >(__u))

6.727.1 Detailed Description

One of the comparison functors.

6.727.2 Member Typedef Documentation

first_argument_type

```
typedef void std::binary_function< void, void, bool >::first_argument_type
```
`first_argument_type` is the type of the first argument

result_type

```
typedef bool std::binary_function< void, void, bool >::result_type
```
`result_type` is the return type

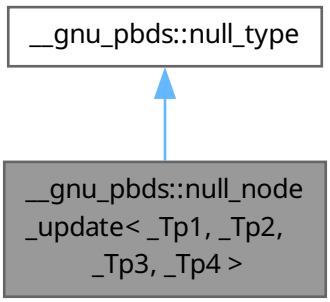
second_argument_type

```
typedef void std::binary_function< void, void, bool >::second_argument_type
```
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- stl_function.h

6.728 __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 > Struct Template Reference

```
#include <tag_and_trait.hpp>
```
Inheritance diagram for __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >:



6.728.1 Detailed Description

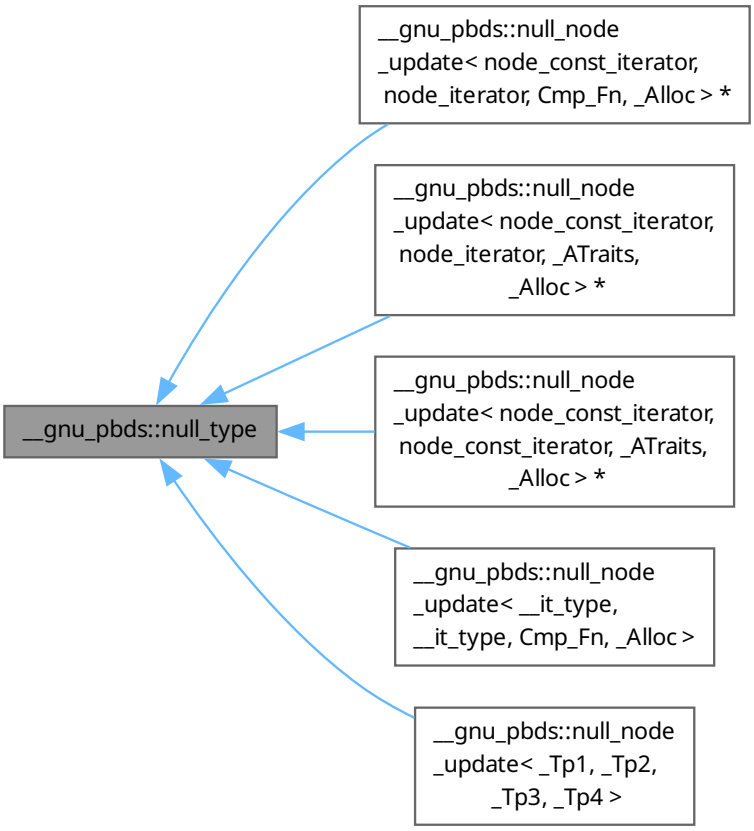
template<typename _Tp1, typename _Tp2, typename _Tp3, typename _Tp4>
struct __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >

A null node updator, indicating that no node updates are required.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.729 __gnu_pbds::null_type Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::null_type:



6.729.1 Detailed Description

Represents no type, or absence of type, for template tricks.
In a mapped-policy, indicates that an associative container is a set.
In a list-update policy, indicates that each link does not need metadata.
In a hash policy, indicates that the combining hash function is actually a ranged hash function.
In a probe policy, indicates that the combining probe function is actually a ranged probe function.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.730 std::experimental::fundamentals_v1::nullopt_t Struct Reference

`#include <optional>`

Public Types

- enum class **_Construct** { **_Token** }

Public Member Functions

- constexpr **nullopt_t** (_Construct)

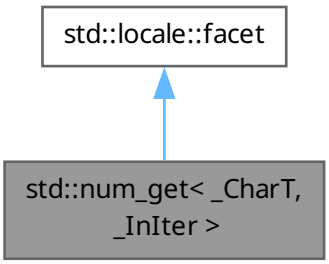
6.730.1 Detailed Description

Tag type to disengage optional objects.
The documentation for this struct was generated from the following file:

- experimental/optional

6.731 std::num_get< _CharT, _InIter > Class Template Reference

`#include <locale_facets.h>`
Inheritance diagram for std::num_get< _CharT, _InIter >:



Public Types

- typedef _CharT char_type
- typedef _InIter iter_type

Public Member Functions

- num_get (size_t __refs=0)
- template<typename _ValueT>
  _InIter **_M_extract_int** (_InIter __beg, _InIter __end, ios_base &__io, ios_base::iostate &__err, _ValueT &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, bool &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, void *&__v) const

- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned short &__v) const

- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned int &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long long &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long long &__v) const

- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, float &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, double &__v) const
- iter_type get (iter_type __in, iter_type __end, ios_base &__io, ios_base::iostate &__err, long double &__v) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~num_get ()
- iter_type **_M_extract_float** (iter_type, iter_type, ios_base &, ios_base::iostate &, string &) const
- template<typename _ValueT>
  iter_type **_M_extract_int** (iter_type, iter_type, ios_base &, ios_base::iostate &, _ValueT &) const
- template<typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, int >::__type **_M_find** (const _CharT2 *, size_t _↩ _len, _CharT2 __c) const
- template<typename _CharT2>
  __gnu_cxx::__enable_if<!__is_char< _CharT2 >::__value, int >::__type **_M_find** (const _CharT2 *__zero, size_t __len, _CharT2 __c) const

- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, bool &) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, long &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned short &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned int &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, long long &__v) const
- virtual iter_type do_get (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, unsigned long long &__v) const
- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, float &) const
- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, double &) const
- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, long double &) const
- virtual iter_type do_get (iter_type, iter_type, ios_base &, ios_base::iostate &, void *&) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.731.1 Detailed Description

template<typename _CharT, typename _InIter>
class std::num_get< _CharT, _InIter >

Primary class template num_get.
This facet encapsulates the code to parse and return a number from a string. It is used by the istream numeric extraction operators.
The num_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num_get facet.

6.731.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _InIter>
typedef _CharT std::num_get< _CharT, _InIter >::char_type
```

Public typedefs.

iter_type

```
template<typename _CharT, typename _InIter>
typedef _InIter std::num_get< _CharT, _InIter >::iter_type
```

Public typedefs.

6.731.3 Constructor & Destructor Documentation

num_get()

```
template<typename _CharT, typename _InIter>
std::num_get< _CharT, _InIter >::num_get (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

~num_get()

```
template<typename _CharT, typename _InIter>
virtual std::num_get< _CharT, _InIter >::~num_get ()  [inline], [protected], [virtual]
```

Destructor.

6.731.4 Member Function Documentation

do_get() [1/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
```

```
            ios_base::iostate & __err,
            long & __v) const  [inline], [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

do_get() [2/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            long long & __v) const  [inline], [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

do_get() [3/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned int & __v) const  [inline], [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

do_get() [4/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned long & __v) const  [inline], [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

do_get() [5/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned long long & __v) const  [inline], [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

do_get() [6/11]

```
template<typename _CharT, typename _InIter>
virtual iter_type std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned short & __v) const  [inline], [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

do_get() [7/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            bool & __v) const  [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.

__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

References std::ios_base::_M_getloc(), std::ios_base::boolalpha, std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::flags(), and std::ios_base::goodbit.
Referenced by std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get(), and std::num_get< _CharT, istreambuf_iterator< _CharT, _Traits > >::get().

do_get() [8/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            double & __v) const  [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::ios_base::eofbit, and std::basic_string< _CharT, _Traits, _Alloc >::r

do_get() [9/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            float & __v) const  [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::ios_base::eofbit, and std::basic_string< _CharT, _Traits, _Alloc >::r

do_get() [10/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            long double & __v) const  [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.

__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

References std::basic_string< _CharT, _Traits, _Alloc >::c_str(), std::ios_base::eofbit, and std::basic_string< _CharT, _Traits, _Alloc >::r

do_get() [11/11]

```
template<typename _CharT, typename _InIter>
_InIter std::num_get< _CharT, _InIter >::do_get (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            void *& __v) const  [protected], [virtual]
```

Numeric parsing.
Parses the input stream into the variable *v*. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__beg	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

References std::ios_base::basefield, std::ios_base::flags(), and std::ios_base::hex.

get() [1/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
```

```
            ios_base::iostate & __err,
            bool & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the bool *v*. It does so by calling num_get::do_get().
If ios_base::boolalpha is set, attempts to read ctype<CharT>::truename() or ctype<CharT>::falsename(). Sets *v* to true or false if successful. Sets err to ios_base::failbit if reading the string fails. Sets err to ios_base::eofbit if the stream is emptied.
If ios_base::boolalpha is not set, proceeds as with reading a long, except if the value is 1, sets *v* to true, if the value is 0, sets *v* to false, and otherwise set err to ios_base::failbit.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

Referenced by std::basic_iostream< char_type, traits_type >::operator>>(), and std::basic_istream< _CharT, _Traits >::operator>>().

get() [2/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            double & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.
The decimal point character used is numpunct::decimal_point(). Digit grouping is interpreted according to numpunct::↩ grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.

__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [3/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            float & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.
The decimal point character used is numpunct::decimal_point(). Digit grouping is interpreted according to numpunct::↩ grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered.
Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [4/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            long & __v) const  [inline]
```

Numeric parsing.

Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
Parsing is affected by the flag settings in *io*.
The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [5/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            long double & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
The input characters are parsed like the scanf g specifier. The matching type length modifier is also used.
The decimal point character used is numpunct::decimal_point(). Digit grouping is interpreted according to numpunct::↩ grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.

__v	Value to format and insert.

Returns

Iterator after reading.

get() [6/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            long long & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
Parsing is affected by the flag settings in *io*.
The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [7/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
```

```
            ios_base::iostate & __err,
            unsigned int & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
Parsing is affected by the flag settings in *io*.
The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [8/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned long & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
Parsing is affected by the flag settings in *io*.
The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [9/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned long long & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
Parsing is affected by the flag settings in *io*.
The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [10/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            unsigned short & __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the integral variable *v*. It does so by calling num_get::do_get().
Parsing is affected by the flag settings in *io*.
The basic parse is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, parses like the scanf o specifier. Else if equal to ios_base::hex, parses like X specifier. Else if basefield equal to 0, parses like the i specifier. Otherwise, parses like d for signed and u for unsigned types. The matching type length modifier is also used.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered.
Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

get() [11/11]

```
template<typename _CharT, typename _InIter>
iter_type std::num_get< _CharT, _InIter >::get (
            iter_type __in,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            void *& __v) const  [inline]
```

Numeric parsing.
Parses the input stream into the pointer variable *v*. It does so by calling num_get::do_get().
The input characters are parsed like the scanf p specifier.
Digit grouping is interpreted according to numpunct::grouping() and numpunct::thousands_sep(). If the pattern of digit groups isn't consistent, sets err to ios_base::failbit.
Note that the digit grouping effect for pointers is a bit ambiguous in the standard and shouldn't be relied on. See DR 344.

If parsing the string yields a valid value for *v*, *v* is set. Otherwise, sets err to ios_base::failbit and leaves *v* unaltered. Sets err to ios_base::eofbit if the stream is emptied.

Parameters

__in	Start of input stream.
__end	End of input stream.
__io	Source of locale and flags.
__err	Error flags to set.
__v	Value to format and insert.

Returns

Iterator after reading.

6.731.5 Member Data Documentation

id

```
template<typename _CharT, typename _InIter>
locale::id std::num_get< _CharT, _InIter >::id  [static]
```

Numpunct facet id.

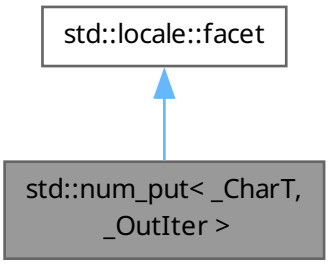
The documentation for this class was generated from the following files:

- locale_facets.h
- locale_facets.tcc

6.732 std::num_put< _CharT, _OutIter > Class Template Reference

```
#include <locale_facets.h>
```

Inheritance diagram for std::num_put< _CharT, _OutIter >:



Public Types

- typedef _CharT char_type
- typedef _OutIter iter_type

Public Member Functions

- num_put (size_t __refs=0)
- template<typename _ValueT>
  _OutIter **_M_insert_float** (_OutIter __s, ios_base &__io, _CharT __fill, char __mod, _ValueT __v) const
- template<typename _ValueT>
  _OutIter **_M_insert_int** (_OutIter __s, ios_base &__io, _CharT __fill, _ValueT __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, bool __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const void *__v) const

- iter_type put (iter_type __s, ios_base &__io, char_type __fill, long __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, unsigned long __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, long long __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, unsigned long long __v) const

- iter_type put (iter_type __s, ios_base &__io, char_type __fill, double __v) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, long double __v) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~num_put ()
- void **_M_group_float** (const char *__grouping, size_t __grouping_size, char_type __sep, const char_type *__p, char_type *__new, char_type *__cs, int &__len) const
- void **_M_group_int** (const char *__grouping, size_t __grouping_size, char_type __sep, ios_base &__io, char_type *__new, char_type *__cs, int &__len) const
- template<typename _ValueT>
  iter_type **_M_insert_float** (iter_type, ios_base &__io, char_type __fill, char __mod, _ValueT __v) const
- template<typename _ValueT>
  iter_type **_M_insert_int** (iter_type, ios_base &__io, char_type __fill, _ValueT __v) const
- void **_M_pad** (char_type __fill, streamsize __w, ios_base &__io, char_type *__new, const char_type *__cs, int &__len) const

- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, bool __v) const
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, long __v) const
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, unsigned long __v) const
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, long long __v) const
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, unsigned long long __v) const
- virtual iter_type do_put (iter_type, ios_base &, char_type, double) const
- virtual iter_type do_put (iter_type, ios_base &, char_type, long double) const
- virtual iter_type do_put (iter_type, ios_base &, char_type, const void *) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.732.1 Detailed Description

template<typename _CharT, typename _OutIter>
class std::num_put< _CharT, _OutIter >

Primary class template num_put.
This facet encapsulates the code to convert a number to a string. It is used by the ostream numeric insertion operators. The num_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the num_put facet.

6.732.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _OutIter>
typedef _CharT std::num_put< _CharT, _OutIter >::char_type
```

Public typedefs.

iter_type

```
template<typename _CharT, typename _OutIter>
typedef _OutIter std::num_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

6.732.3 Constructor & Destructor Documentation

num_put()

```
template<typename _CharT, typename _OutIter>
std::num_put< _CharT, _OutIter >::num_put (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

~num_put()

```
template<typename _CharT, typename _OutIter>
virtual std::num_put< _CharT, _OutIter >::~num_put ()  [inline], [protected], [virtual]
```

Destructor.

6.732.4 Member Function Documentation

do_put() [1/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            bool __v) const  [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

References std::ios_base::_M_getloc(), std::ios_base::adjustfield, std::ios_base::boolalpha, std::ios_base::flags(), std::ios_base::left, and std::ios_base::width().
Referenced by std::num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > >::put(), std::num_put< _CharT, ostreambuf_iterator< _C
std::num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > >::put(), std::num_put< _CharT, ostreambuf_iterator< _CharT, _Traits >
std::num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > >::put(), std::num_put< _CharT, ostreambuf_iterator< _CharT, _Traits >
std::num_put< _CharT, ostreambuf_iterator< _CharT, _Traits > >::put(), and std::num_put< _CharT, ostreambuf_iterator< _CharT, _Tra

do_put() [2/8]

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            long __v) const  [inline], [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.

_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

do_put() [3/8]

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            long long __v) const  [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

do_put() [4/8]

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            unsigned long __v) const  [inline], [protected], [virtual]
```

Numeric formatting.

These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

do_put() [5/8]

```
template<typename _CharT, typename _OutIter>
virtual iter_type std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            unsigned long long __v) const  [inline], [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

do_put() [6/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const void * __v) const  [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

References std::ios_base::basefield, std::ios_base::flags(), std::ios_base::hex, std::ios_base::showbase, and std::ios_base::uppercase.

do_put() [7/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            double __v) const  [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

do_put() [8/8]

```
template<typename _CharT, typename _OutIter>
_OutIter std::num_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            long double __v) const  [protected], [virtual]
```

Numeric formatting.
These functions do the work of formatting numeric values and inserting them into a stream. This function is a hook for derived classes to change the value returned.

Parameters

_↩ _*s*	Stream to write to.
_↩ _*io*	Source of locale and flags.
_↩ _*fill*	Char_type to use for filling.
_↩ _*v*	Value to format and insert.

Returns

Iterator after writing.

put() [1/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            bool __v) const  [inline]
```

Numeric formatting.
Formats the boolean *v* and inserts it into a stream. It does so by calling num_put::do_put().
If ios_base::boolalpha is set, writes ctype<CharT>::truename() or ctype<CharT>::falsename(). Otherwise formats *v* as an int.

Parameters

_↩ _*s*	Stream to write to.
_↩ _*io*	Source of locale and flags.
_↩ _*fill*	Char_type to use for filling.
_↩ _*v*	Value to format and insert.

Returns

Iterator after writing.

Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

put() [2/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const void * __v) const  [inline]
```

Numeric formatting.
Formats the pointer value *v* and inserts it into a stream. It does so by calling num_put::do_put().
This function formats *v* as an unsigned long with ios_base::hex and ios_base::showbase set.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

put() [3/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            double __v) const  [inline]
```

Numeric formatting.
Formats the floating point value *v* and inserts it into a stream. It does so by calling num_put::do_put().
Formatting is affected by the flag settings in *io*.
The basic format is affected by the value of io.flags() & ios_base::floatfield. If equal to ios_base::fixed, formats like the printf f specifier. Else if equal to ios_base::scientific, formats like e or E with ios_base::uppercase unset or set respectively. Otherwise, formats like g or G depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like g or G.
The output precision is given by io.precision(). This precision is capped at numeric_limits::digits10 + 2 (different for double and long double). The default precision is 6.
If ios_base::showpos is set, '+' is output before positive values. If ios_base::showpoint is set, a decimal point will always be output.

The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().
If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_↩ base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

_↩ _*s*	Stream to write to.
_↩ _*io*	Source of locale and flags.
_↩ _*fill*	Char_type to use for filling.
_↩ _*v*	Value to format and insert.

Returns

Iterator after writing.

put() [4/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            long __v) const  [inline]
```

Numeric formatting.
Formats the integral value *v* and inserts it into a stream. It does so by calling num_put::do_put().
Formatting is affected by the flag settings in *io*.
The basic format is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, formats like the printf o specifier. Else if equal to ios_base::hex, formats like x or X with ios_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.
If ios_base::showpos is set, '+' is output before positive values. If ios_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.
The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().
If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_↩ base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

_↩ _*s*	Stream to write to.
_↩ _*io*	Source of locale and flags.

_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

put() [5/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            long double __v) const  [inline]
```

Numeric formatting.
Formats the floating point value *v* and inserts it into a stream. It does so by calling num_put::do_put().
Formatting is affected by the flag settings in *io*.
The basic format is affected by the value of io.flags() & ios_base::floatfield. If equal to ios_base::fixed, formats like the printf f specifier. Else if equal to ios_base::scientific, formats like e or E with ios_base::uppercase unset or set respectively. Otherwise, formats like g or G depending on uppercase. Note that if both fixed and scientific are set, the effect will also be like g or G.
The output precision is given by io.precision(). This precision is capped at numeric_limits::digits10 + 2 (different for double and long double). The default precision is 6.
If ios_base::showpos is set, '+' is output before positive values. If ios_base::showpoint is set, a decimal point will always be output.
The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().
If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_↩ base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

_↩ _s	Stream to write to.
_↩ _io	Source of locale and flags.
_↩ _fill	Char_type to use for filling.
_↩ _v	Value to format and insert.

Returns

Iterator after writing.

put() [6/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            long long __v) const  [inline]
```

Numeric formatting.
Formats the integral value *v* and inserts it into a stream. It does so by calling num_put::do_put().
Formatting is affected by the flag settings in *io*.
The basic format is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, formats like the printf o specifier. Else if equal to ios_base::hex, formats like x or X with ios_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.
If ios_base::showpos is set, '+' is output before positive values. If ios_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.
The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().
If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_↩ base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

_↩ _*s*	Stream to write to.
_↩ _*io*	Source of locale and flags.
_↩ _*fill*	Char_type to use for filling.
_↩ _*v*	Value to format and insert.

Returns

Iterator after writing.

put() [7/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            unsigned long __v) const  [inline]
```

Numeric formatting.
Formats the integral value *v* and inserts it into a stream. It does so by calling num_put::do_put().
Formatting is affected by the flag settings in *io*.
The basic format is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, formats like the printf o specifier. Else if equal to ios_base::hex, formats like x or X with ios_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.

If ios_base::showpos is set, '+' is output before positive values. If ios_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.
The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().
If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_↩ base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

_↩ _*s*	Stream to write to.
_↩ _*io*	Source of locale and flags.
_↩ _*fill*	Char_type to use for filling.
_↩ _*v*	Value to format and insert.

Returns

Iterator after writing.

put() [8/8]

```
template<typename _CharT, typename _OutIter>
iter_type std::num_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            unsigned long long __v) const  [inline]
```

Numeric formatting.
Formats the integral value *v* and inserts it into a stream. It does so by calling num_put::do_put().
Formatting is affected by the flag settings in *io*.
The basic format is affected by the value of io.flags() & ios_base::basefield. If equal to ios_base::oct, formats like the printf o specifier. Else if equal to ios_base::hex, formats like x or X with ios_base::uppercase unset or set respectively. Otherwise, formats like d, ld, lld for signed and u, lu, llu for unsigned values. Note that if both oct and hex are set, neither will take effect.
If ios_base::showpos is set, '+' is output before positive values. If ios_base::showbase is set, '0' precedes octal values (except 0) and '0[xX]' precedes hex values.
The decimal point character used is numpunct::decimal_point(). Thousands separators are inserted according to numpunct::grouping() and numpunct::thousands_sep().
If io.width() is non-zero, enough *fill* characters are inserted to make the result at least that wide. If (io.flags() & ios_↩ base::adjustfield) == ios_base::left, result is padded at the end. If ios_base::internal, then padding occurs immediately after either a '+' or '-' or after '0x' or '0X'. Otherwise, padding occurs at the beginning.

Parameters

_↩ _*s*	Stream to write to.

_↩ _*io*	Source of locale and flags.
_↩ _*fill*	Char_type to use for filling.
_↩ _*v*	Value to format and insert.

Returns

Iterator after writing.

6.732.5 Member Data Documentation

id

```
template<typename _CharT, typename _OutIter>
locale::id std::num_put< _CharT, _OutIter >::id  [static]
```

Numpunct facet id.

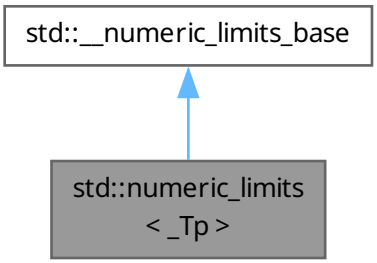
The documentation for this class was generated from the following files:

- locale_facets.h
- locale_facets.tcc

6.733 std::numeric_limits< _Tp > Struct Template Reference

```
#include <limits>
```

Inheritance diagram for std::numeric_limits< _Tp >:



Static Public Member Functions

- static constexpr _Tp denorm_min () noexcept
- static constexpr _Tp epsilon () noexcept
- static constexpr _Tp infinity () noexcept
- static constexpr _Tp lowest () noexcept
- static constexpr _Tp max () noexcept
- static constexpr _Tp min () noexcept

- static constexpr _Tp quiet_NaN () noexcept
- static constexpr _Tp round_error () noexcept
- static constexpr _Tp signaling_NaN () noexcept

Static Public Attributes

- static constexpr int digits
- static constexpr int digits10
- static constexpr float_denorm_style has_denorm
- static constexpr bool has_denorm_loss
- static constexpr bool has_infinity
- static constexpr bool has_quiet_NaN
- static constexpr bool has_signaling_NaN
- static constexpr bool is_bounded
- static constexpr bool is_exact
- static constexpr bool is_iec559
- static constexpr bool is_integer
- static constexpr bool is_modulo
- static constexpr bool is_signed
- static constexpr bool is_specialized
- static constexpr int max_digits10
- static constexpr int max_exponent
- static constexpr int max_exponent10
- static constexpr int min_exponent
- static constexpr int min_exponent10
- static constexpr int radix
- static constexpr float_round_style round_style
- static constexpr bool tinyness_before
- static constexpr bool traps

6.733.1 Detailed Description

template<typename _Tp>
struct std::numeric_limits< _Tp >

Properties of fundamental types.
This class allows a program to obtain information about the representation of a fundamental type on a given platform. For non-fundamental types, the functions will return 0 and the data members will all be `false`.

6.733.2 Member Function Documentation

denorm_min()

```
template<typename _Tp>
constexpr _Tp std::numeric_limits< _Tp >::denorm_min ()  [inline], [static], [constexpr], [noexcept]
```

The minimum positive denormalized value. For types where `has_denorm` is false, this is the minimum positive normalized value.

epsilon()

```
template<typename _Tp>
constexpr _Tp std::numeric_limits< _Tp >::epsilon ()  [inline], [static], [constexpr], [noexcept]
```

The *machine epsilon:* the difference between 1 and the least value greater than 1 that is representable.
Referenced by std::binomial_distribution< _IntType >::operator()(), std::poisson_distribution< _IntType >::operator()(), and std::operator<<().

infinity()

```
template<typename _Tp>
constexpr _Tp std::numeric_limits< _Tp >::infinity () [inline], [static], [constexpr], [noexcept]
```

The representation of positive infinity, if `has_infinity`.

lowest()

```
template<typename _Tp>
constexpr _Tp std::numeric_limits< _Tp >::lowest () [inline], [static], [constexpr], [noexcept]
```

A finite value x such that there is no other finite value y where y < x.

Referenced by std::cauchy_distribution< _RealType >::min(), std::extreme_value_distribution< _RealType >::min(), std::normal_distribution< _RealType >::min(), and std::student_t_distribution< _RealType >::min().

max()

```
template<typename _Tp>
constexpr _Tp std::numeric_limits< _Tp >::max () [inline], [static], [constexpr], [noexcept]
```

The maximum finite value.

Referenced by __gnu_parallel::__parallel_random_shuffle_drs(), std::bernoulli_distribution::max(), std::cauchy_distribution< _RealType >
std::chi_squared_distribution< _RealType >::max(), std::exponential_distribution< _RealType >::max(), std::extreme_value_distribution<
std::fisher_f_distribution< _RealType >::max(), std::gamma_distribution< _RealType >::max(), std::geometric_distribution< _IntType >:
std::lognormal_distribution< _RealType >::max(), std::negative_binomial_distribution< _IntType >::max(), std::normal_distribution< _Re
std::poisson_distribution< _IntType >::max(), std::student_t_distribution< _RealType >::max(), std::weibull_distribution< _RealType >::
std::tr2::dynamic_bitset< _WordT, _Alloc >::max_size(), std::binomial_distribution< _IntType >::operator()(), std::independent_bits_engi
std::poisson_distribution< _IntType >::operator()(), and std::operator<<().

min()

```
template<typename _Tp>
constexpr _Tp std::numeric_limits< _Tp >::min () [inline], [static], [constexpr], [noexcept]
```

The minimum finite value, or for floating types with denormalization, the minimum positive normalized value.

Referenced by std::bernoulli_distribution::min().

quiet_NaN()

```
template<typename _Tp>
constexpr _Tp std::numeric_limits< _Tp >::quiet_NaN () [inline], [static], [constexpr], [noexcept]
```

The representation of a quiet Not a Number, if `has_quiet_NaN`.

round_error()

```
template<typename _Tp>
constexpr _Tp std::numeric_limits< _Tp >::round_error () [inline], [static], [constexpr], [noexcept]
```

The maximum rounding error measurement (see LIA-1).

signaling_NaN()

```
template<typename _Tp>
constexpr _Tp std::numeric_limits< _Tp >::signaling_NaN () [inline], [static], [constexpr],
[noexcept]
```

The representation of a signaling Not a Number, if `has_signaling_NaN`.

6.733.3 Member Data Documentation

digits

```
int std::__numeric_limits_base::digits  [static], [constexpr], [inherited]
```

The number of `radix` digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of `radix` digits in the mantissa.
Referenced by std::generate_canonical(), std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >::operator()(), and std::linear_congruential_engine< _UIntType, __a, __c, __m >::seed().

digits10

```
int std::__numeric_limits_base::digits10  [static], [constexpr], [inherited]
```

The number of base 10 digits that can be represented without change.

has_denorm

```
float_denorm_style std::__numeric_limits_base::has_denorm  [static], [constexpr], [inherited]
```

See std::float_denorm_style for more information.

has_denorm_loss

```
bool std::__numeric_limits_base::has_denorm_loss  [static], [constexpr], [inherited]
```

True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

```
bool std::__numeric_limits_base::has_infinity  [static], [constexpr], [inherited]
```

True if the type has a representation for positive infinity.

has_quiet_NaN

```
bool std::__numeric_limits_base::has_quiet_NaN  [static], [constexpr], [inherited]
```

True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

```
bool std::__numeric_limits_base::has_signaling_NaN  [static], [constexpr], [inherited]
```

True if the type has a representation for a signaling Not a Number.

is_bounded

```
bool std::__numeric_limits_base::is_bounded  [static], [constexpr], [inherited]
```

True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

```
bool std::__numeric_limits_base::is_exact  [static], [constexpr], [inherited]
```

True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

```
bool std::__numeric_limits_base::is_iec559  [static], [constexpr], [inherited]
```

True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr], [inherited]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr], [inherited]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr], [inherited]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr], [inherited]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr], [inherited]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr], [inherited]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr], [inherited]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr], [inherited]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr], [inherited]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr], [inherited]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

Referenced by std::generate_canonical().

round_style

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr], [inherited]
```

See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr], [inherited]
```

True if tininess is detected before rounding. (see IEC 559)

traps

```
bool std::__numeric_limits_base::traps [static], [constexpr], [inherited]
```

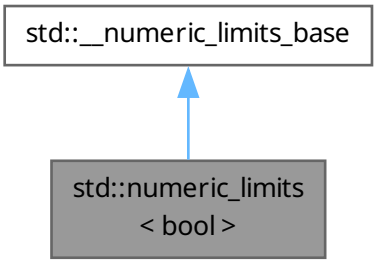
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.734 std::numeric_limits< bool > Struct Reference

```
#include <limits>
```

Inheritance diagram for std::numeric_limits< bool >:



Static Public Member Functions

- static constexpr bool **denorm_min** () noexcept
- static constexpr bool **epsilon** () noexcept
- static constexpr bool **infinity** () noexcept
- static constexpr bool **lowest** () noexcept
- static constexpr bool **max** () noexcept
- static constexpr bool **min** () noexcept
- static constexpr bool **quiet_NaN** () noexcept
- static constexpr bool **round_error** () noexcept
- static constexpr bool **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.734.1 Detailed Description

numeric_limits<bool> specialization.

6.734.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or * on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

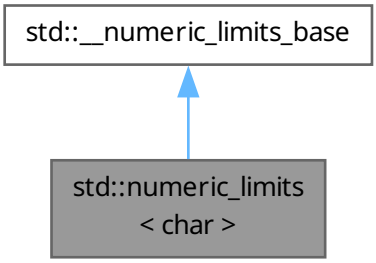
traps

`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.735 std::numeric_limits< char > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< char >:



Static Public Member Functions

- static constexpr char **denorm_min** () noexcept
- static constexpr char **epsilon** () noexcept
- static constexpr char **infinity** () noexcept
- static constexpr char **lowest** () noexcept
- static constexpr char **max** () noexcept
- static constexpr char **min** () noexcept
- static constexpr char **quiet_NaN** () noexcept
- static constexpr char **round_error** () noexcept
- static constexpr char **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.735.1 Detailed Description

numeric_limits<char> specialization.

6.735.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

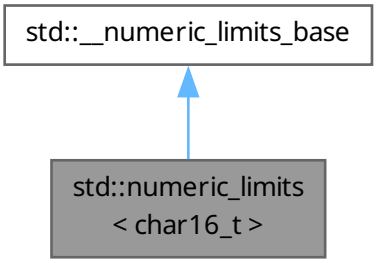
traps

bool std::__numeric_limits_base::traps [static], [constexpr]
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.736 std::numeric_limits< char16_t > Struct Reference

#include <limits>
Inheritance diagram for std::numeric_limits< char16_t >:



Static Public Member Functions

- static constexpr char16_t **denorm_min** () noexcept
- static constexpr char16_t **epsilon** () noexcept
- static constexpr char16_t **infinity** () noexcept
- static constexpr char16_t **lowest** () noexcept
- static constexpr char16_t **max** () noexcept
- static constexpr char16_t **min** () noexcept
- static constexpr char16_t **quiet_NaN** () noexcept
- static constexpr char16_t **round_error** () noexcept
- static constexpr char16_t **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.736.1 Detailed Description

numeric_limits<char16_t> specialization.

6.736.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

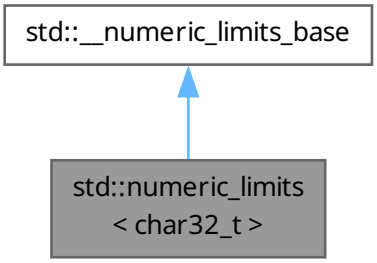
traps

bool std::__numeric_limits_base::traps [static], [constexpr]
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.737 std::numeric_limits< char32_t > Struct Reference

#include <limits>
Inheritance diagram for std::numeric_limits< char32_t >:



Static Public Member Functions

- static constexpr char32_t **denorm_min** () noexcept
- static constexpr char32_t **epsilon** () noexcept
- static constexpr char32_t **infinity** () noexcept
- static constexpr char32_t **lowest** () noexcept
- static constexpr char32_t **max** () noexcept
- static constexpr char32_t **min** () noexcept
- static constexpr char32_t **quiet_NaN** () noexcept
- static constexpr char32_t **round_error** () noexcept
- static constexpr char32_t **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.737.1 Detailed Description

numeric_limits<char32_t> specialization.

6.737.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

```
float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
```
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

```
bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
```
True if tininess is detected before rounding. (see IEC 559)

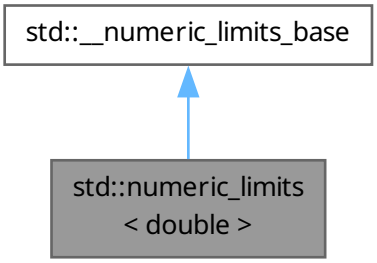
traps

```
bool std::__numeric_limits_base::traps [static], [constexpr]
```
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.738 std::numeric_limits< double > Struct Reference

```
#include <limits>
```
Inheritance diagram for std::numeric_limits< double >:



Static Public Member Functions

- static constexpr double **denorm_min** () noexcept
- static constexpr double **epsilon** () noexcept
- static constexpr double **infinity** () noexcept
- static constexpr double **lowest** () noexcept
- static constexpr double **max** () noexcept
- static constexpr double **min** () noexcept
- static constexpr double **quiet_NaN** () noexcept
- static constexpr double **round_error** () noexcept
- static constexpr double **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.738.1 Detailed Description

numeric_limits<double> specialization.

6.738.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

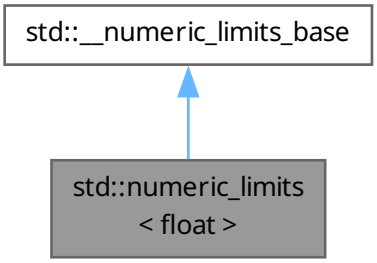
traps

bool std::__numeric_limits_base::traps [static], [constexpr]
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.739 std::numeric_limits< float > Struct Reference

#include <limits>
Inheritance diagram for std::numeric_limits< float >:



Static Public Member Functions

- static constexpr float **denorm_min** () noexcept
- static constexpr float **epsilon** () noexcept
- static constexpr float **infinity** () noexcept
- static constexpr float **lowest** () noexcept
- static constexpr float **max** () noexcept
- static constexpr float **min** () noexcept
- static constexpr float **quiet_NaN** () noexcept
- static constexpr float **round_error** () noexcept
- static constexpr float **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.739.1 Detailed Description

numeric_limits<float> specialization.

6.739.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or * on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

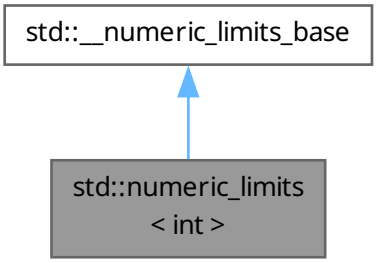
traps

`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.740 std::numeric_limits< int > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< int >:



Static Public Member Functions

- static constexpr int **denorm_min** () noexcept
- static constexpr int **epsilon** () noexcept
- static constexpr int **infinity** () noexcept
- static constexpr int **lowest** () noexcept
- static constexpr int **max** () noexcept
- static constexpr int **min** () noexcept
- static constexpr int **quiet_NaN** () noexcept
- static constexpr int **round_error** () noexcept
- static constexpr int **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.740.1 Detailed Description

numeric_limits<int> specialization.

6.740.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

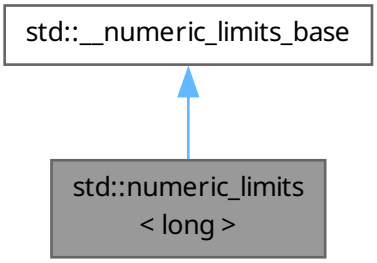
traps

`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.741 std::numeric_limits< long > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< long >:



Static Public Member Functions

- static constexpr long **denorm_min** () noexcept
- static constexpr long **epsilon** () noexcept
- static constexpr long **infinity** () noexcept
- static constexpr long **lowest** () noexcept
- static constexpr long **max** () noexcept
- static constexpr long **min** () noexcept
- static constexpr long **quiet_NaN** () noexcept
- static constexpr long **round_error** () noexcept
- static constexpr long **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.741.1 Detailed Description

numeric_limits<long> specialization.

6.741.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

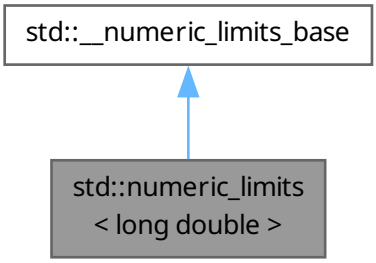
traps

`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.742 std::numeric_limits< long double > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< long double >:



Static Public Member Functions

- static constexpr long double **denorm_min** () noexcept
- static constexpr long double **epsilon** () noexcept
- static constexpr long double **infinity** () noexcept
- static constexpr long double **lowest** () noexcept
- static constexpr long double **max** () noexcept
- static constexpr long double **min** () noexcept
- static constexpr long double **quiet_NaN** () noexcept
- static constexpr long double **round_error** () noexcept
- static constexpr long double **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.742.1 Detailed Description

numeric_limits<long double> specialization.

6.742.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

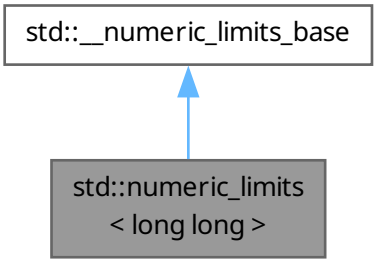
traps

`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.743 std::numeric_limits< long long > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< long long >:



Static Public Member Functions

- static constexpr long long **denorm_min** () noexcept
- static constexpr long long **epsilon** () noexcept
- static constexpr long long **infinity** () noexcept
- static constexpr long long **lowest** () noexcept
- static constexpr long long **max** () noexcept
- static constexpr long long **min** () noexcept
- static constexpr long long **quiet_NaN** () noexcept
- static constexpr long long **round_error** () noexcept
- static constexpr long long **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.743.1 Detailed Description

numeric_limits<long long> specialization.

6.743.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or * on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

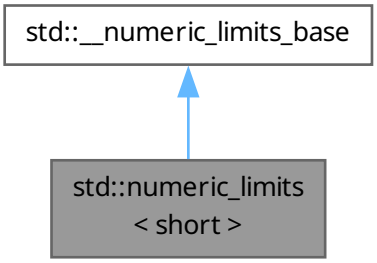
traps

bool std::__numeric_limits_base::traps [static], [constexpr]
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.744 std::numeric_limits< short > Struct Reference

#include <limits>
Inheritance diagram for std::numeric_limits< short >:



Static Public Member Functions

- static constexpr short **denorm_min** () noexcept
- static constexpr short **epsilon** () noexcept
- static constexpr short **infinity** () noexcept
- static constexpr short **lowest** () noexcept
- static constexpr short **max** () noexcept
- static constexpr short **min** () noexcept
- static constexpr short **quiet_NaN** () noexcept
- static constexpr short **round_error** () noexcept
- static constexpr short **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.744.1 Detailed Description

numeric_limits<short> specialization.

6.744.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or * on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

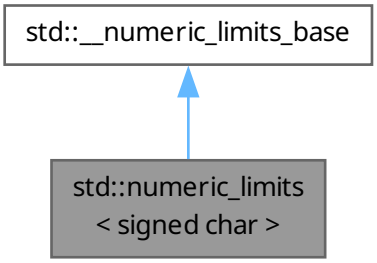
traps

bool std::__numeric_limits_base::traps [static], [constexpr]
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.745 std::numeric_limits< signed char > Struct Reference

#include <limits>
Inheritance diagram for std::numeric_limits< signed char >:



Static Public Member Functions

- static constexpr signed char **denorm_min** () noexcept
- static constexpr signed char **epsilon** () noexcept
- static constexpr signed char **infinity** () noexcept
- static constexpr signed char **lowest** () noexcept
- static constexpr signed char **max** () noexcept
- static constexpr signed char **min** () noexcept
- static constexpr signed char **quiet_NaN** () noexcept
- static constexpr signed char **round_error** () noexcept
- static constexpr signed char **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.745.1 Detailed Description

numeric_limits<signed char> specialization.

6.745.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

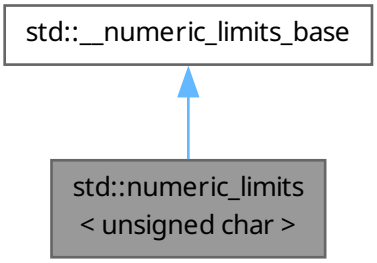
traps

bool std::__numeric_limits_base::traps [static], [constexpr]
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.746 std::numeric_limits< unsigned char > Struct Reference

#include <limits>
Inheritance diagram for std::numeric_limits< unsigned char >:



Static Public Member Functions

- static constexpr unsigned char **denorm_min** () noexcept
- static constexpr unsigned char **epsilon** () noexcept
- static constexpr unsigned char **infinity** () noexcept
- static constexpr unsigned char **lowest** () noexcept
- static constexpr unsigned char **max** () noexcept
- static constexpr unsigned char **min** () noexcept
- static constexpr unsigned char **quiet_NaN** () noexcept
- static constexpr unsigned char **round_error** () noexcept
- static constexpr unsigned char **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.746.1 Detailed Description

numeric_limits<unsigned char> specialization.

6.746.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

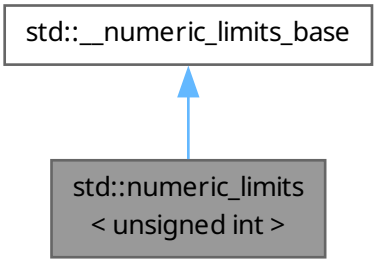
traps

bool std::__numeric_limits_base::traps [static], [constexpr]
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.747 std::numeric_limits< unsigned int > Struct Reference

#include <limits>
Inheritance diagram for std::numeric_limits< unsigned int >:



Static Public Member Functions

- static constexpr unsigned int **denorm_min** () noexcept
- static constexpr unsigned int **epsilon** () noexcept
- static constexpr unsigned int **infinity** () noexcept
- static constexpr unsigned int **lowest** () noexcept
- static constexpr unsigned int **max** () noexcept
- static constexpr unsigned int **min** () noexcept
- static constexpr unsigned int **quiet_NaN** () noexcept
- static constexpr unsigned int **round_error** () noexcept
- static constexpr unsigned int **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.747.1 Detailed Description

numeric_limits<unsigned int> specialization.

6.747.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

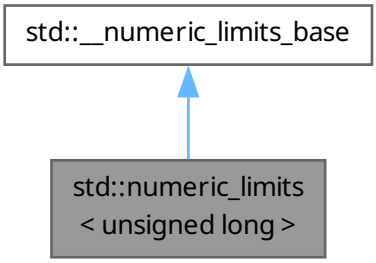
traps

bool std::__numeric_limits_base::traps [static], [constexpr]
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.748 std::numeric_limits< unsigned long > Struct Reference

#include <limits>
Inheritance diagram for std::numeric_limits< unsigned long >:



Static Public Member Functions

- static constexpr unsigned long **denorm_min** () noexcept
- static constexpr unsigned long **epsilon** () noexcept
- static constexpr unsigned long **infinity** () noexcept
- static constexpr unsigned long **lowest** () noexcept
- static constexpr unsigned long **max** () noexcept
- static constexpr unsigned long **min** () noexcept
- static constexpr unsigned long **quiet_NaN** () noexcept
- static constexpr unsigned long **round_error** () noexcept
- static constexpr unsigned long **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.748.1 Detailed Description

numeric_limits<unsigned long> specialization.

6.748.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

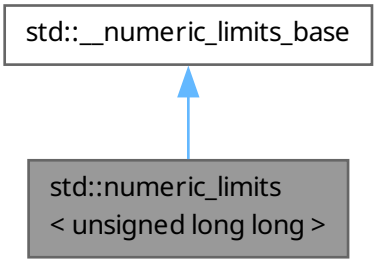
traps

bool std::__numeric_limits_base::traps [static], [constexpr]
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.749 std::numeric_limits< unsigned long long > Struct Reference

#include <limits>
Inheritance diagram for std::numeric_limits< unsigned long long >:



Static Public Member Functions

- static constexpr unsigned long long **denorm_min** () noexcept
- static constexpr unsigned long long **epsilon** () noexcept
- static constexpr unsigned long long **infinity** () noexcept
- static constexpr unsigned long long **lowest** () noexcept
- static constexpr unsigned long long **max** () noexcept
- static constexpr unsigned long long **min** () noexcept
- static constexpr unsigned long long **quiet_NaN** () noexcept
- static constexpr unsigned long long **round_error** () noexcept
- static constexpr unsigned long long **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.749.1 Detailed Description

numeric_limits<unsigned long long> specialization.

6.749.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

`bool std::__numeric_limits_base::is_integer [static], [constexpr]`
True if the type is integer.

is_modulo

`bool std::__numeric_limits_base::is_modulo [static], [constexpr]`
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

`bool std::__numeric_limits_base::is_signed [static], [constexpr]`
True if the type is signed.

is_specialized

`bool std::__numeric_limits_base::is_specialized [static], [constexpr]`
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

`int std::__numeric_limits_base::max_digits10 [static], [constexpr]`
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

`int std::__numeric_limits_base::max_exponent [static], [constexpr]`
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

`int std::__numeric_limits_base::max_exponent10 [static], [constexpr]`
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

`int std::__numeric_limits_base::min_exponent [static], [constexpr]`
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

`int std::__numeric_limits_base::min_exponent10 [static], [constexpr]`
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

`int std::__numeric_limits_base::radix [static], [constexpr]`
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

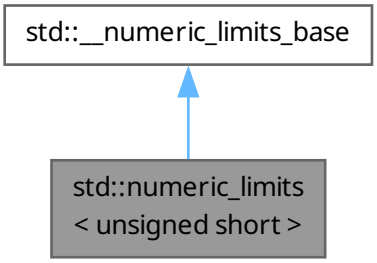
traps

bool std::__numeric_limits_base::traps [static], [constexpr]
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.750 std::numeric_limits< unsigned short > Struct Reference

#include <limits>
Inheritance diagram for std::numeric_limits< unsigned short >:



Static Public Member Functions

- static constexpr unsigned short **denorm_min** () noexcept
- static constexpr unsigned short **epsilon** () noexcept
- static constexpr unsigned short **infinity** () noexcept
- static constexpr unsigned short **lowest** () noexcept
- static constexpr unsigned short **max** () noexcept
- static constexpr unsigned short **min** () noexcept
- static constexpr unsigned short **quiet_NaN** () noexcept
- static constexpr unsigned short **round_error** () noexcept
- static constexpr unsigned short **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.750.1 Detailed Description

numeric_limits<unsigned short> specialization.

6.750.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or * on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

`float_round_style std::__numeric_limits_base::round_style [static], [constexpr]`
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

`bool std::__numeric_limits_base::tinyness_before [static], [constexpr]`
True if tininess is detected before rounding. (see IEC 559)

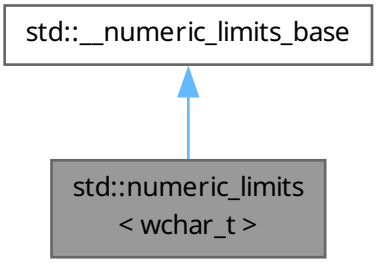
traps

`bool std::__numeric_limits_base::traps [static], [constexpr]`
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.751 std::numeric_limits< wchar_t > Struct Reference

`#include <limits>`
Inheritance diagram for std::numeric_limits< wchar_t >:



Static Public Member Functions

- static constexpr wchar_t **denorm_min** () noexcept
- static constexpr wchar_t **epsilon** () noexcept
- static constexpr wchar_t **infinity** () noexcept
- static constexpr wchar_t **lowest** () noexcept
- static constexpr wchar_t **max** () noexcept
- static constexpr wchar_t **min** () noexcept
- static constexpr wchar_t **quiet_NaN** () noexcept
- static constexpr wchar_t **round_error** () noexcept
- static constexpr wchar_t **signaling_NaN** () noexcept

Static Public Attributes

- static constexpr int **digits**
- static constexpr int digits
- static constexpr int **digits10**
- static constexpr int digits10
- static constexpr float_denorm_style **has_denorm**
- static constexpr float_denorm_style has_denorm
- static constexpr bool **has_denorm_loss**
- static constexpr bool has_denorm_loss
- static constexpr bool **has_infinity**
- static constexpr bool has_infinity
- static constexpr bool **has_quiet_NaN**
- static constexpr bool has_quiet_NaN
- static constexpr bool **has_signaling_NaN**
- static constexpr bool has_signaling_NaN
- static constexpr bool **is_bounded**
- static constexpr bool is_bounded
- static constexpr bool **is_exact**
- static constexpr bool is_exact
- static constexpr bool **is_iec559**
- static constexpr bool is_iec559
- static constexpr bool **is_integer**
- static constexpr bool is_integer
- static constexpr bool **is_modulo**
- static constexpr bool is_modulo
- static constexpr bool **is_signed**
- static constexpr bool is_signed
- static constexpr bool **is_specialized**
- static constexpr bool is_specialized
- static constexpr int **max_digits10**
- static constexpr int max_digits10
- static constexpr int **max_exponent**
- static constexpr int max_exponent
- static constexpr int **max_exponent10**
- static constexpr int max_exponent10
- static constexpr int **min_exponent**
- static constexpr int min_exponent
- static constexpr int **min_exponent10**
- static constexpr int min_exponent10
- static constexpr int **radix**
- static constexpr int radix
- static constexpr float_round_style **round_style**
- static constexpr float_round_style round_style
- static constexpr bool **tinyness_before**
- static constexpr bool tinyness_before
- static constexpr bool **traps**
- static constexpr bool traps

6.751.1 Detailed Description

numeric_limits<wchar_t> specialization.

6.751.2 Member Data Documentation

digits

int std::__numeric_limits_base::digits [static], [constexpr]
The number of radix digits that be represented without change: for integer types, the number of non-sign bits in the mantissa; for floating types, the number of radix digits in the mantissa.

digits10

int std::__numeric_limits_base::digits10 [static], [constexpr]
The number of base 10 digits that can be represented without change.

has_denorm

float_denorm_style std::__numeric_limits_base::has_denorm [static], [constexpr]
See std::float_denorm_style for more information.

has_denorm_loss

bool std::__numeric_limits_base::has_denorm_loss [static], [constexpr]
True if loss of accuracy is detected as a denormalization loss, rather than as an inexact result.

has_infinity

bool std::__numeric_limits_base::has_infinity [static], [constexpr]
True if the type has a representation for positive infinity.

has_quiet_NaN

bool std::__numeric_limits_base::has_quiet_NaN [static], [constexpr]
True if the type has a representation for a quiet (non-signaling) Not a Number.

has_signaling_NaN

bool std::__numeric_limits_base::has_signaling_NaN [static], [constexpr]
True if the type has a representation for a signaling Not a Number.

is_bounded

bool std::__numeric_limits_base::is_bounded [static], [constexpr]
True if the set of values representable by the type is finite. All built-in types are bounded, this member would be false for arbitrary precision types.

is_exact

bool std::__numeric_limits_base::is_exact [static], [constexpr]
True if the type uses an exact representation. All integer types are exact, but not all exact types are integer. For example, rational and fixed-exponent representations are exact but not integer.

is_iec559

bool std::__numeric_limits_base::is_iec559 [static], [constexpr]
True if-and-only-if the type adheres to the IEC 559 standard, also known as IEEE 754. (Only makes sense for floating point types.)

is_integer

bool std::__numeric_limits_base::is_integer [static], [constexpr]
True if the type is integer.

is_modulo

bool std::__numeric_limits_base::is_modulo [static], [constexpr]
True if the type is *modulo*. A type is modulo if, for any operation involving +, -, or $*$ on values of that type whose result would fall outside the range [min(),max()], the value returned differs from the true value by an integer multiple of max() - min() + 1. On most machines, this is false for floating types, true for unsigned integers, and true for signed integers. See PR22200 about signed integers.

is_signed

bool std::__numeric_limits_base::is_signed [static], [constexpr]
True if the type is signed.

is_specialized

bool std::__numeric_limits_base::is_specialized [static], [constexpr]
This will be true for all fundamental types (which have specializations), and false for everything else.

max_digits10

int std::__numeric_limits_base::max_digits10 [static], [constexpr]
The number of base 10 digits required to ensure that values which differ are always differentiated.

max_exponent

int std::__numeric_limits_base::max_exponent [static], [constexpr]
The maximum positive integer such that `radix` raised to the power of (one less than that integer) is a representable finite floating point number.

max_exponent10

int std::__numeric_limits_base::max_exponent10 [static], [constexpr]
The maximum positive integer such that 10 raised to that power is in the range of representable finite floating point numbers.

min_exponent

int std::__numeric_limits_base::min_exponent [static], [constexpr]
The minimum negative integer such that `radix` raised to the power of (one less than that integer) is a normalized floating point number.

min_exponent10

int std::__numeric_limits_base::min_exponent10 [static], [constexpr]
The minimum negative integer such that 10 raised to that power is in the range of normalized floating point numbers.

radix

int std::__numeric_limits_base::radix [static], [constexpr]
For integer types, specifies the base of the representation. For floating types, specifies the base of the exponent representation.

round_style

float_round_style std::__numeric_limits_base::round_style [static], [constexpr]
See std::float_round_style for more information. This is only meaningful for floating types; integer types will all be round_toward_zero.

tinyness_before

bool std::__numeric_limits_base::tinyness_before [static], [constexpr]
True if tininess is detected before rounding. (see IEC 559)

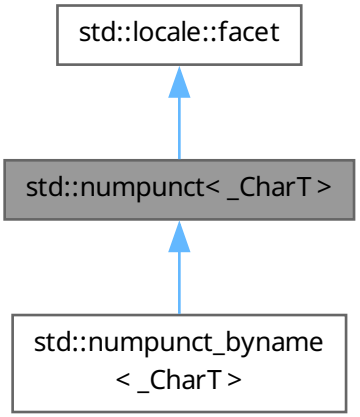
traps

bool std::__numeric_limits_base::traps [static], [constexpr]
True if trapping is implemented for this type.
The documentation for this struct was generated from the following file:

- limits

6.752 std::numpunct< _CharT > Class Template Reference

#include <locale_facets.h>
Inheritance diagram for std::numpunct< _CharT >:



Public Types

- typedef __numpunct_cache< _CharT > **__cache_type**

- typedef _CharT char_type
- typedef basic_string< _CharT > string_type

Public Member Functions

- numpunct (__c_locale __cloc, size_t __refs=0)
- numpunct (__cache_type *__cache, size_t __refs=0)
- numpunct (size_t __refs=0)
- char_type decimal_point () const
- string_type falsename () const
- string grouping () const
- char_type thousands_sep () const
- string_type truename () const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~numpunct ()
- void **_M_initialize_numpunct** (__c_locale __cloc)
- void **_M_initialize_numpunct** (__c_locale __cloc)
- void **_M_initialize_numpunct** (__c_locale __cloc=0)
- virtual char_type do_decimal_point () const
- virtual string_type do_falsename () const
- virtual string do_grouping () const
- virtual char_type do_thousands_sep () const
- virtual string_type do_truename () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __cache_type * **_M_data**

6.752.1 Detailed Description

template<typename _CharT>
class std::numpunct< _CharT >

Primary class template numpunct.
This facet stores several pieces of information related to printing and scanning numbers, such as the decimal point character. It takes a template parameter specifying the char type. The numpunct facet is used by streams for many I/O operations involving numbers.
The numpunct template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from a numpunct facet.

6.752.2 Member Typedef Documentation

char_type

```
template<typename _CharT>
typedef _CharT std::numpunct< _CharT >::char_type
```

Public typedefs.

string_type

```
template<typename _CharT>
typedef basic_string<_CharT> std::numpunct< _CharT >::string_type
```

Public typedefs.

6.752.3 Constructor & Destructor Documentation

numpunct() [1/3]

```
template<typename _CharT>
std::numpunct< _CharT >::numpunct (
            size_t __refs = 0)  [inline], [explicit]
```

Numpunct constructor.

Parameters

__refs	Refcount to pass to the base class.

References std::locale::facet::facet().

numpunct() [2/3]

```
template<typename _CharT>
std::numpunct< _CharT >::numpunct (
            __cache_type * __cache,
            size_t __refs = 0)  [inline], [explicit]
```

Internal constructor. Not for general use.
This is a constructor for use by the library itself to set up the predefined locale facets.

Parameters

__cache	__numpunct_cache object.
__refs	Refcount to pass to the base class.

References std::locale::facet::facet().

numpunct() [3/3]

```
template<typename _CharT>
std::numpunct< _CharT >::numpunct (
            __c_locale __cloc,
            size_t __refs = 0)  [inline], [explicit]
```

Internal constructor. Not for general use.
This is a constructor for use by the library itself to set up new locales.

Parameters

__cloc	The C locale.
__refs	Refcount to pass to the base class.

References std::locale::facet::facet().

~numpunct()

```
template<typename _CharT>
virtual std::numpunct< _CharT >::~numpunct ()  [protected], [virtual]
```

Destructor.

6.752.4 Member Function Documentation

decimal_point()

```
template<typename _CharT>
char_type std::numpunct< _CharT >::decimal_point () const  [inline]
```

Return decimal point character.

This function returns a char_type to use as a decimal point. It does so by returning returning numpunct<char_type>::↩ do_decimal_point().

Returns

char_type representing a decimal point.

References do_decimal_point().

do_decimal_point()

```
template<typename _CharT>
virtual char_type std::numpunct< _CharT >::do_decimal_point () const  [inline], [protected],
[virtual]
```

Return decimal point character.

Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Referenced by decimal_point().

do_falsename()

```
template<typename _CharT>
virtual string_type std::numpunct< _CharT >::do_falsename () const  [inline], [protected], [virtual]
```

Return string representation of bool false.

Returns a string_type containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of false.

Referenced by falsename().

do_grouping()

```
template<typename _CharT>
virtual string std::numpunct< _CharT >::do_grouping () const  [inline], [protected], [virtual]
```

Return grouping specification.

Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

grouping() for details.

Returns

String representing grouping specification.

Referenced by grouping().

do_thousands_sep()

```
template<typename _CharT>
virtual char_type std::numpunct< _CharT >::do_thousands_sep () const  [inline], [protected],
[virtual]
```

Return thousands separator character.

Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Referenced by thousands_sep().

do_truename()

```
template<typename _CharT>
virtual string_type std::numpunct< _CharT >::do_truename () const  [inline], [protected], [virtual]
```

Return string representation of bool true.

Returns a string_type containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of true.

Referenced by truename().

falsename()

```
template<typename _CharT>
string_type std::numpunct< _CharT >::falsename () const  [inline]
```

Return string representation of bool false.

This function returns a string_type containing the text representation for false bool variables. It does so by calling numpunct<char_type>::do_falsename().

Returns

string_type representing printed form of false.

References do_falsename().

grouping()

```
template<typename _CharT>
string std::numpunct< _CharT >::grouping () const  [inline]
```

Return grouping specification.
This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.
Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.
For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.
The string is returned by calling numpunct<char_type>::do_grouping().

Returns

string representing grouping specification.

References do_grouping().

thousands_sep()

```
template<typename _CharT>
char_type std::numpunct< _CharT >::thousands_sep () const  [inline]
```

Return thousands separator character.
This function returns a char_type to use as a thousands separator. It does so by returning returning numpunct<char_type>::do_thousands_sep().

Returns

char_type representing a thousands separator.

References do_thousands_sep().

truename()

```
template<typename _CharT>
string_type std::numpunct< _CharT >::truename () const  [inline]
```

Return string representation of bool true.
This function returns a string_type containing the text representation for true bool variables. It does so by calling numpunct<char_type>::do_truename().

Returns

string_type representing printed form of true.

References do_truename().

6.752.5 Member Data Documentation

id

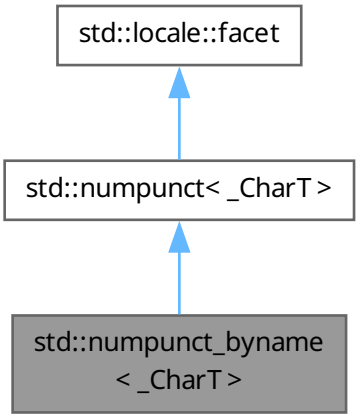
```
template<typename _CharT>
locale::id std::numpunct< _CharT >::id  [static]
```

Numpunct facet id.
The documentation for this class was generated from the following file:

- locale_facets.h

6.753 std::numpunct_byname< _CharT > Class Template Reference

`#include <locale_facets.h>`
Inheritance diagram for std::numpunct_byname< _CharT >:



Public Types

- typedef __numpunct_cache< _CharT > **__cache_type**
- typedef _CharT **char_type**
- typedef basic_string< _CharT > **string_type**

Public Member Functions

- **numpunct_byname** (const char *__s, size_t __refs=0)
- **numpunct_byname** (const string &__s, size_t __refs=0)
- char_type decimal_point () const
- string_type falsename () const
- string grouping () const
- char_type thousands_sep () const
- string_type truename () const

Static Public Attributes

- static locale::id id

Protected Member Functions

- void **_M_initialize_numpunct** (__c_locale __cloc)
- void **_M_initialize_numpunct** (__c_locale __cloc)
- void **_M_initialize_numpunct** (__c_locale __cloc=0)
- virtual char_type do_decimal_point () const
- virtual string_type do_falsename () const
- virtual string do_grouping () const

- virtual char_type do_thousands_sep () const
- virtual string_type do_truename () const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

Protected Attributes

- __cache_type * **_M_data**

6.753.1 Detailed Description

template<typename _CharT>
class std::numpunct_byname< _CharT >

class numpunct_byname [22.2.3.2].

6.753.2 Member Function Documentation

decimal_point()

```
template<typename _CharT>
char_type std::numpunct< _CharT >::decimal_point () const  [inline], [inherited]
```

Return decimal point character.
This function returns a char_type to use as a decimal point. It does so by returning returning numpunct<char_type>::↩ do_decimal_point().

Returns

char_type representing a decimal point.

References do_decimal_point().

do_decimal_point()

```
template<typename _CharT>
virtual char_type std::numpunct< _CharT >::do_decimal_point () const  [inline], [protected],
[virtual], [inherited]
```

Return decimal point character.
Returns a char_type to use as a decimal point. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a decimal point.

Referenced by decimal_point().

do_falsename()

```
template<typename _CharT>
virtual string_type std::numpunct< _CharT >::do_falsename () const  [inline], [protected], [virtual],
[inherited]
```

Return string representation of bool false.
Returns a string_type containing the text representation for false bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of false.

Referenced by falsename().

do_grouping()

```
template<typename _CharT>
virtual string std::numpunct< _CharT >::do_grouping () const  [inline], [protected], [virtual],
[inherited]
```

Return grouping specification.
Returns a string representing groupings for the integer part of a number. This function is a hook for derived classes to change the value returned.

See also

grouping() for details.

Returns

String representing grouping specification.

Referenced by grouping().

do_thousands_sep()

```
template<typename _CharT>
virtual char_type std::numpunct< _CharT >::do_thousands_sep () const  [inline], [protected],
[virtual], [inherited]
```

Return thousands separator character.
Returns a char_type to use as a thousands separator. This function is a hook for derived classes to change the value returned.

Returns

char_type representing a thousands separator.

Referenced by thousands_sep().

do_truename()

```
template<typename _CharT>
virtual string_type std::numpunct< _CharT >::do_truename () const  [inline], [protected], [virtual],
[inherited]
```

Return string representation of bool true.
Returns a string_type containing the text representation for true bool variables. This function is a hook for derived classes to change the value returned.

Returns

string_type representing printed form of true.

Referenced by truename().

falsename()

```
template<typename _CharT>
string_type std::numpunct< _CharT >::falsename () const  [inline], [inherited]
```

Return string representation of bool false.

This function returns a string_type containing the text representation for false bool variables. It does so by calling numpunct<char_type>::do_falsename().

Returns

string_type representing printed form of false.

References do_falsename().

grouping()

```
template<typename _CharT>
string std::numpunct< _CharT >::grouping () const  [inline], [inherited]
```

Return grouping specification.

This function returns a string representing groupings for the integer part of a number. Groupings indicate where thousands separators should be inserted in the integer part of a number.

Each char in the return string is interpret as an integer rather than a character. These numbers represent the number of digits in a group. The first char in the string represents the number of digits in the least significant group. If a char is negative, it indicates an unlimited number of digits for the group. If more chars from the string are required to group a number, the last char is used repeatedly.

For example, if the grouping() returns "\003\002" and is applied to the number 123456789, this corresponds to 12,34,56,789. Note that if the string was "32", this would put more than 50 digits into the least significant group if the character set is ASCII.

The string is returned by calling numpunct<char_type>::do_grouping().

Returns

string representing grouping specification.

References do_grouping().

thousands_sep()

```
template<typename _CharT>
char_type std::numpunct< _CharT >::thousands_sep () const  [inline], [inherited]
```

Return thousands separator character.

This function returns a char_type to use as a thousands separator. It does so by returning returning numpunct<char↩ _type>::do_thousands_sep().

Returns

char_type representing a thousands separator.

References do_thousands_sep().

truename()

```
template<typename _CharT>
string_type std::numpunct< _CharT >::truename () const  [inline], [inherited]
```

Return string representation of bool true.

This function returns a string_type containing the text representation for true bool variables. It does so by calling numpunct<char_type>::do_truename().

Returns

string_type representing printed form of true.

References do_truename().

6.753.3 Member Data Documentation

id

```
template<typename _CharT>
locale::id std::numpunct< _CharT >::id  [static], [inherited]
```

Numpunct facet id.

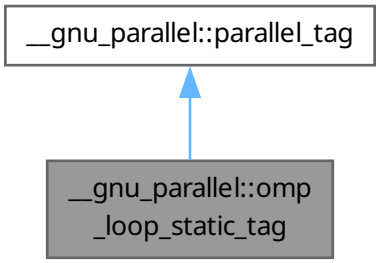
The documentation for this class was generated from the following file:

- locale_facets.h

6.754 __gnu_parallel::omp_loop_static_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::omp_loop_static_tag:



Public Member Functions

- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

6.754.1 Detailed Description

Recommends parallel execution using OpenMP static load-balancing at compile time.

6.754.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

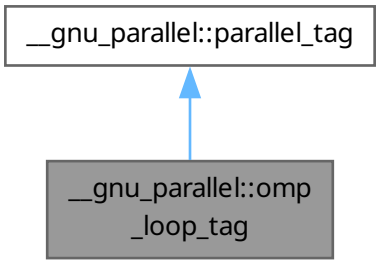
The documentation for this struct was generated from the following file:

- tags.h

6.755 __gnu_parallel::omp_loop_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::omp_loop_tag:



Public Member Functions

- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

6.755.1 Detailed Description

Recommends parallel execution using OpenMP dynamic load-balancing at compile time.

6.755.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

6.756 std::once_flag Struct Reference

```
#include <mutex>
```

Public Member Functions

- once_flag (const once_flag &)=delete
- once_flag & operator= (const once_flag &)=delete

Friends

- template<typename _Callable, typename... _Args>
  void call_once (once_flag &__once, _Callable &&__f, _Args &&... __args)

6.756.1 Detailed Description

Flag type used by std::call_once.

6.756.2 Constructor & Destructor Documentation

once_flag()

```
std::once_flag::once_flag (
            const once_flag & )  [delete]
```

Deleted copy constructor.

6.756.3 Member Function Documentation

operator=()

```
once_flag & std::once_flag::operator= (
            const once_flag & )  [delete]
```

Deleted assignment operator.

6.756.4 Friends And Related Symbol Documentation

call_once

```
template<typename _Callable, typename...  _Args>
void call_once (
            once_flag & __once,
            _Callable && __f,
            _Args &&...  __args)  [friend]
```

Invoke a callable and synchronize with other calls using the same flag.
References std::__invoke(), and std::forward().
The documentation for this struct was generated from the following file:

- mutex

6.757 std::experimental::fundamentals_v1::optional< _Tp > Class Template Reference

`#include <optional>`

Public Types

- using **value_type**

Public Member Functions

- template<typename _Up = _Tp, enable_if_t< __and_< __not_< is_same< optional< _Tp >, decay_t< _Up > > >, is_constructible< _Tp, _Up && >, __not_< is_convertible< _Up &&, _Tp > > >::value, bool > = false>
  constexpr **optional** (_Up &&__t)
- template<typename _Up = _Tp, enable_if_t< __and_< __not_< is_same< optional< _Tp >, decay_t< _Up > > >, is_constructible< _Tp, _Up && >, is_convertible< _Up &&, _Tp > >::value, bool > = true>
  constexpr **optional** (_Up &&__t)
- template<typename _Up, enable_if_t< __and_< __not_< is_same< _Tp, _Up > >, is_constructible< _Tp, const _Up & >, __not_< is_convertible< const _Up &, _Tp > >, __not_< __converts_from_optional< _Tp, _Up > > >::value, bool > = false>
  constexpr **optional** (const optional< _Up > &__t)
- template<typename _Up, enable_if_t< __and_< __not_< is_same< _Tp, _Up > >, is_constructible< _Tp, const _Up & >, is_convertible< const _Up &, _Tp >, __not_< __converts_from_optional< _Tp, _Up > > >::value, bool > = true>
  constexpr **optional** (const optional< _Up > &__t)
- template<typename _Up, enable_if_t< __and_< __not_< is_same< _Tp, _Up > >, is_constructible< _Tp, _Up && >, __not_< is_convertible< _Up &&, _Tp >, __not_< __converts_from_optional< _Tp, _Up > > >::value, bool > = false>
  constexpr **optional** (optional< _Up > &&__t)
- template<typename _Up, enable_if_t< __and_< __not_< is_same< _Tp, _Up > >, is_constructible< _Tp, _Up && >, is_convertible< _Up &&, _Tp >, __not_< __converts_from_optional< _Tp, _Up > > >::value, bool > = true>
  constexpr **optional** (optional< _Up > &&__t)
- template<typename... _Args>
  enable_if_t< is_constructible< _Tp, _Args &&... >::value > **emplace** (_Args &&... __args)
- template<typename _Up, typename... _Args>
  enable_if_t< is_constructible< _Tp, initializer_list< _Up > &, _Args &&... >::value > **emplace** (initializer_list< _Up > __il, _Args &&... __args)
- constexpr **operator bool** () const noexcept
- constexpr _Tp & **operator*** () &
- constexpr _Tp && **operator*** () &&
- constexpr const _Tp & **operator*** () const &
- constexpr const _Tp && **operator*** () const &&
- _Tp * **operator->** ()
- constexpr const _Tp * **operator->** () const
- template<typename _Up = _Tp>
  enable_if_t< __and_< __not_< is_same< optional< _Tp >, decay_t< _Up > > >, is_constructible< _Tp, _Up >, __not_< __and_< is_scalar< _Tp >, is_same< _Tp, decay_t< _Up > > >, is_assignable< _Tp &, _Up > >::value, optional & > **operator=** (_Up &&__u)
- template<typename _Up>
  enable_if_t< __and_< __not_< is_same< _Tp, _Up > >, is_constructible< _Tp, const _Up & >, is_assignable< _Tp &, _Up >, __not_< __converts_from_optional< _Tp, _Up > >, __not_< __assigns_from_optional< _Tp, _Up > > >::value, optional & > **operator=** (const optional< _Up > &__u)

- optional & **operator=** (nullopt_t) noexcept
- template<typename _Up>
  enable_if_t< __and_< __not_< is_same< _Tp, _Up > >, is_constructible< _Tp, _Up >, is_assignable< _Tp &, _Up >, __not_< __converts_from_optional< _Tp, _Up > >, __not_< __assigns_from_optional< _Tp, _Up > > >::value, optional & > **operator=** (optional< _Up > &&__u)
- void **swap** (optional &__other) noexcept(is_nothrow_move_constructible< _Tp >() &&__is_nothrow_↩ swappable< _Tp >::value)
- constexpr _Tp & **value** () &
- constexpr _Tp && **value** () &&
- constexpr const _Tp & **value** () const &
- constexpr const _Tp && **value** () const &&
- template<typename _Up>
  _Tp **value_or** (_Up &&__u) &&
- template<typename _Up>
  constexpr _Tp **value_or** (_Up &&__u) const &

6.757.1 Detailed Description

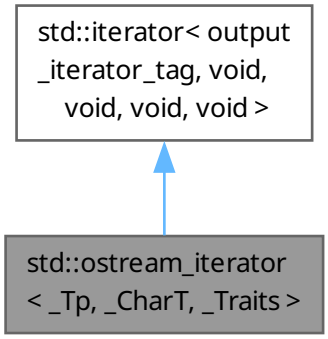
template<typename _Tp>
class std::experimental::fundamentals_v1::optional< _Tp >

Class template for optional values.
The documentation for this class was generated from the following file:

- experimental/optional

6.758 std::ostream_iterator< _Tp, _CharT, _Traits > Class Template Reference

`#include <stream_iterator.h>`
Inheritance diagram for std::ostream_iterator< _Tp, _CharT, _Traits >:



Public Types

- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference

- typedef void value_type

- using difference_type
- typedef _CharT char_type
- typedef _Traits traits_type
- typedef basic_ostream< _CharT, _Traits > ostream_type

Public Member Functions

- ostream_iterator (const ostream_iterator &__obj) noexcept
- ostream_iterator (ostream_type &__s) noexcept
- ostream_iterator (ostream_type &__s, const _CharT *__c) noexcept
- ostream_iterator & **operator*** () noexcept
- ostream_iterator & **operator++** () noexcept
- ostream_iterator & **operator++** (int) noexcept
- ostream_iterator & operator= (const _Tp &__value)
- ostream_iterator & **operator=** (const ostream_iterator &)=default

6.758.1 Detailed Description

template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
class std::ostream_iterator< _Tp, _CharT, _Traits >

Provides output iterator semantics for streams.
This class provides an iterator to write to an ostream. The type Tp is the only type written by this iterator and there must be an operator<<(Tp) defined.

Template Parameters

_Tp	The type to write to the ostream.
_CharT	The ostream char_type.
_Traits	The ostream char_traits.

6.758.2 Member Typedef Documentation

char_type

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef _CharT std::ostream_iterator< _Tp, _CharT, _Traits >::char_type
```

Public typedef.

difference_type

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
using std::ostream_iterator< _Tp, _CharT, _Traits >::difference_type
```

Public typedef.

iterator_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::↩
iterator_category  [inherited]
```

One of the tag types.

ostream_type

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef basic_ostream<_CharT, _Traits> std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_↩
type
```
Public typedef.

pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer  [inherited]
```
This type represents a pointer-to-value_type.

reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference  [inherited]
```
This type represents a reference-to-value_type.

traits_type

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
typedef _Traits std::ostream_iterator< _Tp, _CharT, _Traits >::traits_type
```
Public typedef.

value_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type  [inherited]
```
The type "pointed to" by the iterator.

6.758.3 Constructor & Destructor Documentation

ostream_iterator() [1/3]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
            ostream_type & __s)  [inline], [noexcept]
```
Construct from an ostream.
References std::__addressof().
Referenced by ostream_iterator().

ostream_iterator() [2/3]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
            ostream_type & __s,
            const _CharT * __c)  [inline], [noexcept]
```
Construct from an ostream.
The delimiter string *c* is written to the stream after every Tp written to the stream. The delimiter is not copied, and thus must not be destroyed while this iterator is in use.

Parameters

_↩ _s	Underlying ostream to write to.
_↩ _c	CharT delimiter string to insert.

References std::__addressof().

ostream_iterator() [3/3]

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
std::ostream_iterator< _Tp, _CharT, _Traits >::ostream_iterator (
            const ostream_iterator< _Tp, _CharT, _Traits > & __obj)  [inline], [noexcept]
```

Copy constructor.
References ostream_iterator().

6.758.4 Member Function Documentation

operator=()

```
template<typename _Tp, typename _CharT = char, typename _Traits = char_traits<_CharT>>
ostream_iterator & std::ostream_iterator< _Tp, _CharT, _Traits >::operator= (
            const _Tp & __value)  [inline]
```

Writes *value* to underlying ostream using operator<<. If constructed with delimiter string, writes delimiter to ostream.
The documentation for this class was generated from the following file:

- stream_iterator.h

6.759 std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits > Class Template Reference

```
#include <iterator>
```

Public Types

- typedef _CharT **char_type**
- typedef void **difference_type**
- typedef output_iterator_tag **iterator_category**
- typedef basic_ostream< _CharT, _Traits > **ostream_type**
- typedef void **pointer**
- typedef void **reference**
- typedef _Traits **traits_type**
- typedef void **value_type**

Public Member Functions

- **ostream_joiner** (ostream_type &__os, _DelimT &&__delimiter) noexcept(is_nothrow_move_constructible_v< ↩ _DelimT >)
- **ostream_joiner** (ostream_type &__os, const _DelimT &__delimiter) noexcept(is_nothrow_copy_constructible↩ _v< _DelimT >)
- ostream_joiner & **operator*** () noexcept
- ostream_joiner & **operator++** () noexcept
- ostream_joiner & **operator++** (int) noexcept
- template<typename _Tp>
  ostream_joiner & **operator=** (const _Tp &__value)

6.759.1 Detailed Description

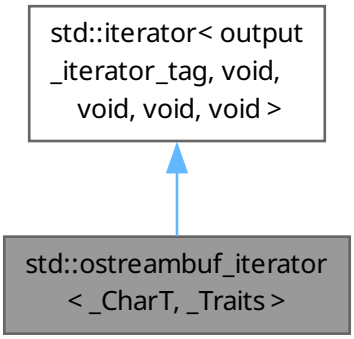
template<typename _DelimT, typename _CharT = char, typename _Traits = char_traits<_CharT>>
class std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits >

Output iterator that inserts a delimiter between elements.
The documentation for this class was generated from the following file:

- experimental/iterator

6.760 std::ostreambuf_iterator< _CharT, _Traits > Class Template Reference

`#include <streambuf_iterator.h>`
Inheritance diagram for std::ostreambuf_iterator< _CharT, _Traits >:



Public Types

- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

- using difference_type
- typedef _CharT char_type
- typedef _Traits traits_type
- typedef basic_streambuf< _CharT, _Traits > streambuf_type
- typedef basic_ostream< _CharT, _Traits > ostream_type

Public Member Functions

- ostreambuf_iterator (ostream_type &__s) noexcept
- ostreambuf_iterator (streambuf_type *__s) noexcept
- ostreambuf_iterator & **_M_put** (const _CharT *__ws, streamsize __len)
- bool failed () const noexcept
- ostreambuf_iterator & operator* ()

- ostreambuf_iterator & operator++ ()
- ostreambuf_iterator & operator++ (int)
- ostreambuf_iterator & operator= (_CharT __c)

Friends

- template<typename _CharT2>
  __gnu_cxx::__enable_if< __is_char< _CharT2 >::__value, ostreambuf_iterator< _CharT2 > >::__type **copy** (istreambuf_iterator< _CharT2 >, istreambuf_iterator< _CharT2 >, ostreambuf_iterator< _CharT2 >)

6.760.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::ostreambuf_iterator< _CharT, _Traits >

Provides output iterator semantics for streambufs.

6.760.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::ostreambuf_iterator< _CharT, _Traits >::char_type
```

Public typedefs.

difference_type

```
template<typename _CharT, typename _Traits>
using std::ostreambuf_iterator< _CharT, _Traits >::difference_type
```

Public typedefs.

iterator_category

```
typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::↩
iterator_category [inherited]
```

One of the tag types.

ostream_type

```
template<typename _CharT, typename _Traits>
typedef basic_ostream<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::ostream_type
```

Public typedefs.

pointer

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
```

This type represents a pointer-to-value_type.

reference

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
```

This type represents a reference-to-value_type.

streambuf_type

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<_CharT, _Traits> std::ostreambuf_iterator< _CharT, _Traits >::streambuf↩
_type
```
Public typedefs.

traits_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::ostreambuf_iterator< _CharT, _Traits >::traits_type
```
Public typedefs.

value_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type  [inherited]
```
The type "pointed to" by the iterator.

6.760.3 Constructor & Destructor Documentation

ostreambuf_iterator() [1/2]

```
template<typename _CharT, typename _Traits>
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
            ostream_type & __s)  [inline], [noexcept]
```
Construct output iterator from ostream.

ostreambuf_iterator() [2/2]

```
template<typename _CharT, typename _Traits>
std::ostreambuf_iterator< _CharT, _Traits >::ostreambuf_iterator (
            streambuf_type * __s)  [inline], [noexcept]
```
Construct output iterator from streambuf.

6.760.4 Member Function Documentation

failed()

```
template<typename _CharT, typename _Traits>
bool std::ostreambuf_iterator< _CharT, _Traits >::failed () const  [inline], [nodiscard], [noexcept]
```
Return true if previous operator=() failed.
Referenced by std::basic_ostream< _CharT, _Traits >::operator<<().

operator*()

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator* ()  [inline], [nodiscard]
```
Return *this.

operator++() [1/2]

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator++ ()  [inline]
```
Return *this.

operator++() [2/2]

```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator++ (
            int )  [inline]
```

Return ∗this.

operator=()

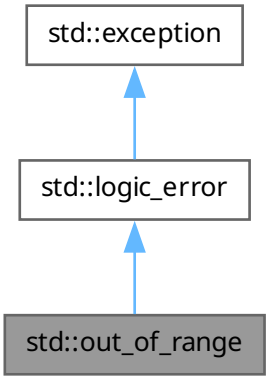
```
template<typename _CharT, typename _Traits>
ostreambuf_iterator & std::ostreambuf_iterator< _CharT, _Traits >::operator= (
            _CharT __c)  [inline]
```

Write character to streambuf. Calls streambuf.sputc().
The documentation for this class was generated from the following files:

- iosfwd.h
- streambuf_iterator.h

6.761 std::out_of_range Class Reference

`#include <stdexcept_except.h>`
Inheritance diagram for std::out_of_range:



Public Member Functions

- **out_of_range** (const char ∗)
- **out_of_range** (const out_of_range &)=default
- **out_of_range** (const string &__arg)
- **out_of_range** (out_of_range &&)=default
- out_of_range & **operator=** (const out_of_range &)=default
- out_of_range & **operator=** (out_of_range &&)=default
- virtual const char ∗ what () const noexcept

6.761.1 Detailed Description

This represents an argument whose value is not within the expected range (e.g., boundary checks in basic_string).

6.761.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
References what().
Referenced by what().
The documentation for this class was generated from the following file:

- stdexcept_except.h

6.762 std::out_ptr_t< _Smart, _Pointer, _Args > Class Template Reference

```
#include <>>
```

Public Member Functions

- **out_ptr_t** (_Smart &__smart, _Args... __args)
- **out_ptr_t** (const out_ptr_t &)=delete
- **operator _Pointer** * () const noexcept
- **operator void** ** () const noexcept

Friends

- template<typename, typename, typename...>
  class **inout_ptr_t**

6.762.1 Detailed Description

template<typename _Smart, typename _Pointer, typename... _Args>
class std::out_ptr_t< _Smart, _Pointer, _Args >

Smart pointer adaptor for functions taking an output pointer parameter.

Template Parameters

_Smart	The type of pointer to adapt.
_Pointer	The type of pointer to convert to.
_Args...	Argument types used when resetting the smart pointer.

Since

C++23

The documentation for this class was generated from the following file:

- out_ptr.h

6.763 std::output_iterator_tag Struct Reference

`#include <stl_iterator_base_types.h>`

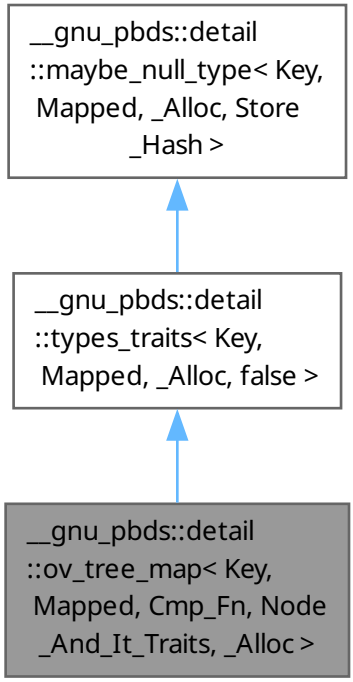
6.763.1 Detailed Description

Marking output iterators.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

6.764 __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference

`#include <ov_tree_map_.hpp>`
Inheritance diagram for __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >:



Classes

- class cond_dtor

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**

- typedef std::pair< size_type, size_type > **comp_hash**
- typedef point_const_iterator **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef ov_tree_tag **container_category**
- typedef _Alloc::difference_type **difference_type**
- typedef point_iterator **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- typedef traits_type::node_update **node_update**
- typedef const_pointer **point_const_iterator**
- typedef pointer **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **ov_tree_map** (const Cmp_Fn &)
- **ov_tree_map** (const Cmp_Fn &, const node_update &)
- **ov_tree_map** (const ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It>
  void **copy_from_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator it)
- bool **erase** (key_const_reference)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference r_key)
- point_const_iterator **find** (key_const_reference r_key) const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const

- std::pair< point_iterator, bool > **insert** (const_reference r_value)
- void **join** (ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **lower_bound** (key_const_reference r_key)
- point_const_iterator **lower_bound** (key_const_reference r_key) const
- size_type **max_size** () const
- node_iterator node_begin ()
- node_const_iterator node_begin () const
- node_iterator node_end ()
- node_const_iterator node_end () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- size_type **size** () const
- void **split** (key_const_reference, ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference r_key)
- point_const_iterator **upper_bound** (key_const_reference r_key) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

6.764.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩ Alloc>
class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Ordered-vector tree associative-container.

6.764.2 Member Function Documentation

node_begin() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::ov_tree_map<
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin ()  [inline]
```

Returns a node_iterator corresponding to the node at the root of the tree.

node_begin() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::ov_tree_m
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin () const  [inline]
```

Returns a const node_iterator corresponding to the node at the root of the tree.

node_end() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::ov_tree_map<
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ()  [inline]
```

Returns a node_iterator corresponding to a node just after a leaf of the tree.

node_end() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::ov_tree_n
Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end () const  [inline]
```

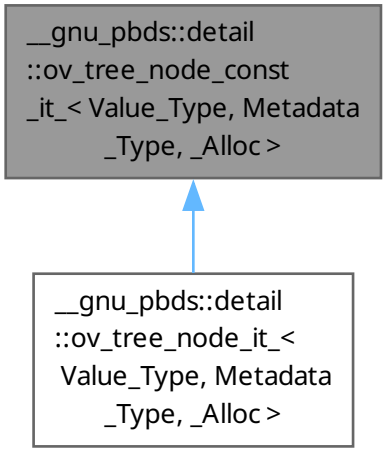
Returns a const node_iterator corresponding to a node just after a leaf of the tree.
The documentation for this class was generated from the following file:

- ov_tree_map_.hpp

6.765 __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference

```
#include <node_iterators.hpp>
```

Inheritance diagram for __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >:



Public Types

- typedef rebind_traits< _Alloc, typenameremove_const< Value_Type >::type >::const_pointer **const_reference**
- typedef trivial_iterator_difference_type **difference_type**
- typedef trivial_iterator_tag **iterator_category**
- typedef rebind_traits< _Alloc, metadata_type >::const_reference **metadata_const_reference**
- typedef Metadata_Type **metadata_type**
- typedef rebind_traits< _Alloc, typenameremove_const< Value_Type >::type >::const_pointer **reference**
- typedef rebind_traits< _Alloc, Value_Type >::const_pointer **value_type**

Public Member Functions

- **ov_tree_node_const_it_** (const_pointer p_nd=0, const_pointer p_begin_nd=0, const_pointer p_end_nd=0, const_metadata_pointer p_metadata=0)
- this_type get_l_child () const

- metadata_const_reference **get_metadata** () const
- this_type get_r_child () const
- bool **operator!=** (const this_type &other) const
- const_reference **operator*** () const
- bool **operator==** (const this_type &other) const

Public Attributes

- pointer **m_p_begin_value**
- pointer **m_p_end_value**
- const_metadata_pointer **m_p_metadata**
- pointer **m_p_value**

Protected Types

- typedef rebind_traits< _Alloc, Metadata_Type >::const_pointer **const_metadata_pointer**
- typedef rebind_traits< _Alloc, Value_Type >::const_pointer **const_pointer**
- typedef rebind_traits< _Alloc, Value_Type >::pointer **pointer**
- typedef ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc > **this_type**

Static Protected Member Functions

- template<typename Ptr>
  static Ptr **mid_pointer** (Ptr p_begin, Ptr p_end)

6.765.1 Detailed Description

template<typename Value_Type, typename Metadata_Type, typename _Alloc>
class __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >

Const node reference.

6.765.2 Member Function Documentation

get_l_child()

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
this_type __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_↩
l_child () const  [inline]
```

Returns the node iterator associated with the left node.

get_r_child()

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
this_type __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >::get_↩
r_child () const  [inline]
```

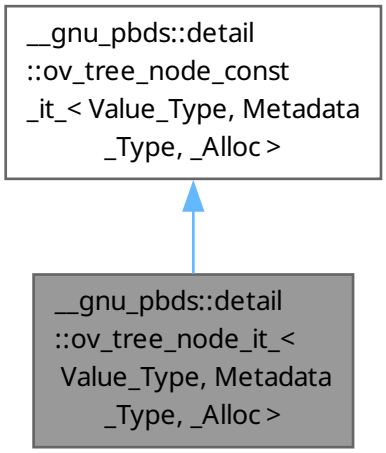
Returns the node iterator associated with the right node.
The documentation for this class was generated from the following file:

- ov_tree_map_/node_iterators.hpp

6.766 __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc > Class Template Reference

`#include <node_iterators.hpp>`

Inheritance diagram for __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >:



Public Types

- typedef rebind_traits< _Alloc, typenameremove_const< Value_Type >::type >::pointer **const_reference**
- typedef trivial_iterator_difference_type **difference_type**
- typedef trivial_iterator_tag **iterator_category**
- typedef rebind_traits< _Alloc, metadata_type >::const_reference **metadata_const_reference**
- typedef Metadata_Type **metadata_type**
- typedef rebind_traits< _Alloc, typenameremove_const< Value_Type >::type >::pointer **reference**
- typedef rebind_traits< _Alloc, Value_Type >::pointer **value_type**

Public Member Functions

- **ov_tree_node_it_** (const_pointer p_nd=0, const_pointer p_begin_nd=0, const_pointer p_end_nd=0, const_↩ metadata_pointer p_metadata=0)
- ov_tree_node_it_ get_l_child () const
- metadata_const_reference **get_metadata** () const
- ov_tree_node_it_ get_r_child () const
- bool **operator!=** (const this_type &other) const
- reference operator* () const
- bool **operator==** (const this_type &other) const

Public Attributes

- pointer **m_p_begin_value**
- pointer **m_p_end_value**
- const_metadata_pointer **m_p_metadata**
- pointer **m_p_value**

Static Protected Member Functions

- template<typename Ptr>
  static Ptr **mid_pointer** (Ptr p_begin, Ptr p_end)

6.766.1 Detailed Description

template<typename Value_Type, typename Metadata_Type, typename _Alloc>
class __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >

Node reference.

6.766.2 Member Function Documentation

get_l_child()

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get↩
_l_child () const  [inline]
```

Returns the node reference associated with the left node.

get_r_child()

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
ov_tree_node_it_ __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::get↩
_r_child () const  [inline]
```

Returns the node reference associated with the right node.

operator*()

```
template<typename Value_Type, typename Metadata_Type, typename _Alloc>
reference __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >::operator* ()
const  [inline]
```

Access.

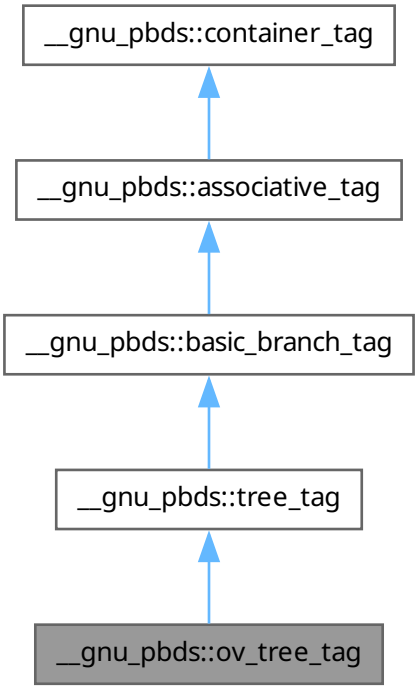
The documentation for this class was generated from the following file:

- ov_tree_map_/node_iterators.hpp

6.767 __gnu_pbds::ov_tree_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::ov_tree_tag:



6.767.1 Detailed Description

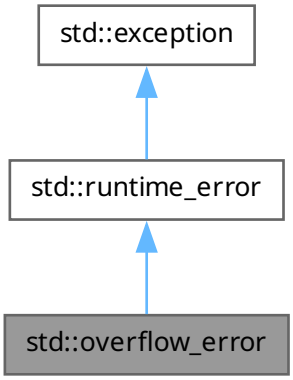
Ordered-vector tree.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.768 std::overflow_error Class Reference

`#include <stdexcept_except.h>`

Inheritance diagram for std::overflow_error:



Public Member Functions

- **overflow_error** (const char *)
- **overflow_error** (const overflow_error &)=default
- **overflow_error** (const string &__arg)
- **overflow_error** (overflow_error &&)=default
- overflow_error & **operator=** (const overflow_error &)=default
- overflow_error & **operator=** (overflow_error &&)=default
- virtual const char * what () const noexcept

6.768.1 Detailed Description

Thrown to indicate arithmetic overflow.

6.768.2 Member Function Documentation

what()

```
virtual const char * std::runtime_error::what () const  [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::experimental::filesystem::v1::filesystem_error, std::filesystem::filesystem_error, and std::ios_base::failure.
References what().
Referenced by what().
The documentation for this class was generated from the following file:

- stdexcept_except.h

6.769 std::owner_less< _Tp > Struct Template Reference

6.769.1 Detailed Description

template<typename _Tp = void>
struct std::owner_less< _Tp >

Primary template owner_less.
The documentation for this struct was generated from the following file:

- bits/shared_ptr.h

6.770 std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > > Struct Template Reference

`#include <shared_ptr.h>`

Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__lhs, const _Tp &__rhs) const noexcept
- bool **operator()** (const _Tp &__lhs, const _Tp1 &__rhs) const noexcept
- bool **operator()** (const _Tp1 &__lhs, const _Tp &__rhs) const noexcept

6.770.1 Detailed Description

template<typename _Tp>
struct std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >

Partial specialization of owner_less for shared_ptr.

6.770.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- experimental/bits/shared_ptr.h

6.771 std::owner_less< shared_ptr< _Tp > > Struct Template Reference

`#include <shared_ptr.h>`

Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__lhs, const _Tp &__rhs) const noexcept
- bool **operator()** (const _Tp &__lhs, const _Tp1 &__rhs) const noexcept
- bool **operator()** (const _Tp1 &__lhs, const _Tp &__rhs) const noexcept

6.771.1 Detailed Description

template<typename _Tp>
struct std::owner_less< shared_ptr< _Tp > >

Partial specialization of owner_less for shared_ptr.

6.771.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

`typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]`
`result_type` is the return type

second_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- bits/shared_ptr.h

6.772 std::owner_less< void > Struct Reference

`#include <shared_ptr.h>`

Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__lhs, const _Tp &__rhs) const noexcept
- bool **operator()** (const _Tp &__lhs, const _Tp1 &__rhs) const noexcept
- bool **operator()** (const _Tp1 &__lhs, const _Tp &__rhs) const noexcept

6.772.1 Detailed Description

Void specialization of owner_less compares either shared_ptr or weak_ptr.

6.772.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type  [inherited]
```

first_argument_type is the type of the first argument

result_type

```
typedef bool std::binary_function< _Tp, _Tp, bool >::result_type  [inherited]
```

result_type is the return type

second_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type  [inherited]
```

second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- bits/shared_ptr.h

6.773 std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > > Struct Template Reference

```
#include <shared_ptr.h>
```

Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__lhs, const _Tp &__rhs) const noexcept
- bool **operator()** (const _Tp &__lhs, const _Tp1 &__rhs) const noexcept
- bool **operator()** (const _Tp1 &__lhs, const _Tp &__rhs) const noexcept

6.773.1 Detailed Description

template<typename _Tp>
struct std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >

Partial specialization of owner_less for weak_ptr.

6.773.2 Member Typedef Documentation

first_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type  [inherited]
```

first_argument_type is the type of the first argument

result_type

typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
result_type is the return type

second_argument_type

typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- experimental/bits/shared_ptr.h

6.774 std::owner_less< weak_ptr< _Tp > > Struct Template Reference

#include <shared_ptr.h>

Public Types

- typedef _Tp first_argument_type
- typedef bool result_type
- typedef _Tp second_argument_type

Public Member Functions

- bool **operator()** (const _Tp &__lhs, const _Tp &__rhs) const noexcept
- bool **operator()** (const _Tp &__lhs, const _Tp1 &__rhs) const noexcept
- bool **operator()** (const _Tp1 &__lhs, const _Tp &__rhs) const noexcept

6.774.1 Detailed Description

template<typename _Tp>
struct std::owner_less< weak_ptr< _Tp > >

Partial specialization of owner_less for weak_ptr.

6.774.2 Member Typedef Documentation

first_argument_type

typedef _Tp std::binary_function< _Tp, _Tp, bool >::first_argument_type [inherited]
first_argument_type is the type of the first argument

result_type

typedef bool std::binary_function< _Tp, _Tp, bool >::result_type [inherited]
result_type is the return type

second_argument_type

typedef _Tp std::binary_function< _Tp, _Tp, bool >::second_argument_type [inherited]
second_argument_type is the type of the second argument
The documentation for this struct was generated from the following file:

- bits/shared_ptr.h

6.775 std::packaged_task< _Res(_ArgTypes...)> Class Template Reference

`#include <future>`

Public Member Functions

- template<typename _Fn, typename = __not_same<_Fn>>
  packaged_task (_Fn &&__fn)
- template<typename _Fn, typename _Alloc, typename = __not_same<_Fn>>
  packaged_task (allocator_arg_t, const _Alloc &__a, _Fn &&__fn)
- **packaged_task** (const packaged_task &)=delete
- **packaged_task** (packaged_task &&__other) noexcept
- future< _Res > **get_future** ()
- void **make_ready_at_thread_exit** (_ArgTypes... __args)
- void **operator()** (_ArgTypes... __args)
- packaged_task & **operator=** (const packaged_task &)=delete
- packaged_task & **operator=** (packaged_task &&__other) noexcept
- void **reset** ()
- void **swap** (packaged_task &__other) noexcept
- bool **valid** () const noexcept

6.775.1 Detailed Description

template<typename _Res, typename... _ArgTypes>
class std::packaged_task< _Res(_ArgTypes...)>

packaged_task
The documentation for this class was generated from the following file:

- future

6.776 __gnu_cxx::pair< _T1, _T2 > Struct Template Reference

`#include <utility>`

Public Types

- typedef _T1 first_type
- typedef _T2 second_type

Public Member Functions

- constexpr pair (const pair &)=default
- constexpr pair (pair &&)=default
- template<typename... _Args1, typename... _Args2>
  constexpr pair (piecewise_construct_t, tuple< _Args1... >, tuple< _Args2... >)
- template<typename _U1, typename _U2>
  constexpr **explicit** (!_S_convertible< _U1 &, _U2 & >()) pair(pair< _U1
- template<typename _U1, typename _U2>
  constexpr _U2 &&constexpr explicit (!_S_convertible< _U1 &, _U2 & >()) pair(pair< _U1
- template<typename _U1 = _T1, typename _U2 = _T2>
  constexpr **explicit** (!_S_convertible< _U1, _U2 >()) pair(_U1 &&
- template<typename _U1 = _T1, typename _U2 = _T2>
  constexpr explicit (!_S_convertible< _U1, _U2 >()) pair(_U1 &&__x

- template<typename _U1, typename _U2>
  constexpr **explicit** (!_S_convertible< _U1, _U2 >()) pair(pair< _U1
- template<typename _U1, typename _U2>
  constexpr _U2 &constexpr explicit (!_S_convertible< _U1, _U2 >()) pair(pair< _U1
- constexpr explicit (!_S_convertible< const _T1 &, const _T2 & >()) pair(const type_identity_t< _T1 > &__x
- template<typename _U1, typename _U2>
  constexpr **explicit** (!_S_convertible< const _U1 &, const _U2 & >()) pair(const pair< _U1
- template<typename _U1, typename _U2>
  constexpr _U2 &&constexpr explicit (!_S_convertible< const _U1 &, const _U2 & >()) pair(const pair< _U1
- template<typename _U1, typename _U2>
  constexpr **explicit** (!_S_convertible< const _U1, const _U2 >()) pair(const pair< _U1
- template<typename _U1, typename _U2>
  constexpr _U2 &constexpr explicit (!_S_convertible< const _U1, const _U2 >()) pair(const pair< _U1
- template<__eligible_pair_like< pair > _UPair>
  constexpr **explicit** (!_S_convertible_from_pair_like< _UPair >()) pair(_UPair &&)=delete
- template<__eligible_pair_like< pair > _UPair>
  constexpr _U2 &&constexpr **explicit** (!_S_convertible_from_pair_like< _UPair >()) pair(_UPair &&__p)
- constexpr explicit (__not_< __and_< __is_implicitly_default_constructible< _T1 >, __is_implicitly_default_↩
  constructible< _T2 > > >()) pair() noexcept(is_nothrow_default_constructible_v< _T1 > &&is_nothrow_↩
  default_constructible_v< _T2 >)
- constexpr _U2 &&constexpr _U2 &__p **noexcept** (_S_nothrow_constructible< _U1 &, _U2 & >())
- constexpr _U2 &constexpr _U2 &&__p **noexcept** (_S_nothrow_constructible< _U1, _U2 >())
- constexpr _U2 &&__y **noexcept** (_S_nothrow_constructible< _U1, _U2 >())
- constexpr const _T2 &__y **noexcept** (_S_nothrow_constructible< const _T1 &, const _T2 & >())
- constexpr _U2 &&constexpr _U2 &__p **noexcept** (_S_nothrow_constructible< const _U1 &, const _U2 & >())
- constexpr _U2 &constexpr _U2 &&__p **noexcept** (_S_nothrow_constructible< const _U1, const _U2 >())
- template<__eligible_pair_like< pair > _UPair>
  constexpr pair & **operator=** (_UPair &&__p)
- template<__eligible_pair_like< pair > _UPair>
  constexpr const pair & **operator=** (_UPair &&__p) const
- pair & **operator=** (const pair &)=delete
- constexpr const pair & operator= (const pair &__p) const
- constexpr pair & operator= (const pair &__p) noexcept(_S_nothrow_assignable< const _T1 &, const _T2 & >())
- template<typename _U1, typename _U2>
  requires (_S_const_assignable<const _U1&, const _U2&>())
  constexpr const pair & operator= (const pair< _U1, _U2 > &__p) const
- template<typename _U1, typename _U2>
  requires (_S_assignable<const _U1&, const _U2&>())
  constexpr pair & operator= (const pair< _U1, _U2 > &__p) noexcept(_S_nothrow_assignable< const _U1 &, const _U2 & >())
- constexpr const pair & operator= (pair &&__p) const
- constexpr pair & operator= (pair &&__p) noexcept(_S_nothrow_assignable< _T1, _T2 >())
- template<typename _U1, typename _U2>
  requires (_S_const_assignable<_U1, _U2>())
  constexpr const pair & operator= (pair< _U1, _U2 > &&__p) const
- template<typename _U1, typename _U2>
  requires (_S_assignable<_U1, _U2>())
  constexpr pair & operator= (pair< _U1, _U2 > &&__p) noexcept(_S_nothrow_assignable< _U1, _U2 >())
- constexpr void **swap** (const pair &__p) const noexcept(__and_v< __is_nothrow_swappable< const _T1 >, _↩
  _is_nothrow_swappable< const _T2 > >)
- constexpr void swap (pair &__p) noexcept(__and_< __is_nothrow_swappable< _T1 >, __is_nothrow_↩
  swappable< _T2 > >::value)

Public Attributes

- _T1 first
- _T2 second

Related Symbols

(Note that these are not member symbols.)

- template<typename _T1, typename _T2>
  pair (_T1, _T2) -> pair< _T1, _T2 >
- template<typename _T1, typename _T2>
  constexpr enable_if< __and_< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2, _U2 > > operator<=> (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)

6.776.1 Detailed Description

template<typename _T1, typename _T2>
struct __gnu_cxx::pair< _T1, _T2 >

Struct holding two objects (or references) of arbitrary type.

Template Parameters

_T1	Type of the `first` member.
_T2	Type of the `second` member.

https://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html

6.776.2 Member Typedef Documentation

first_type

```
template<typename _T1, typename _T2>
typedef _T1 std::pair< _T1, _T2 >::first_type
```

The type of the `first` member.

second_type

```
template<typename _T1, typename _T2>
typedef _T2 std::pair< _T1, _T2 >::second_type
```

The type of the `second` member.

6.776.3 Constructor & Destructor Documentation

pair() [1/2]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::pair (
            const pair< _T1, _T2 > & )  [constexpr], [default]
```

Copy constructor.

pair() [2/2]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::pair (
            pair< _T1, _T2 > && )  [constexpr], [default]
```

Move constructor.

6.776.4 Member Function Documentation

explicit() [1/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &&constexpr std::pair< _T1, _T2 >::explicit (
            !_S_convertible< _U1 &, _U2 & > ())  [constexpr]
```

Converting constructor from a non-const `pair<U1, U2>` lvalue.

explicit() [2/7]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2>
std::pair< _T1, _T2 >::explicit (
            !_S_convertible< _U1, _U2 > ()) &&  [constexpr]
```

Constructor accepting two values of arbitrary types.

explicit() [3/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &constexpr std::pair< _T1, _T2 >::explicit (
            !_S_convertible< _U1, _U2 > ())  [constexpr]
```

Converting constructor from a non-const `pair<U1, U2>` rvalue.

explicit() [4/7]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::explicit (
            !_S_convertible< const _T1 &, const _T2 & > ()) const &  [constexpr]
```

Constructor accepting lvalues of `first_type` and `second_type`.

explicit() [5/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &&constexpr std::pair< _T1, _T2 >::explicit (
            !_S_convertible< const _U1 &, const _U2 & > ()) const  [constexpr]
```

Converting constructor from a const `pair<U1, U2>` lvalue.

explicit() [6/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &constexpr std::pair< _T1, _T2 >::explicit (
            !_S_convertible< const _U1, const _U2 > ()) const  [constexpr]
```

Converting constructor from a const `pair<U1, U2>` rvalue.

explicit() [7/7]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::explicit (
            __not_< __and_< __is_implicitly_default_constructible< _T1 >, __is_implicitly_↩
default_constructible< _T2 > > > ())  [inline], [constexpr], [noexcept]
```

Default constructor.

operator=() [1/8]

```
template<typename _T1, typename _T2>
const pair & std::pair< _T1, _T2 >::operator= (
            const pair< _T1, _T2 > & __p) const  [inline], [constexpr]
```

Copy assignment operator (const).

operator=() [2/8]

```
template<typename _T1, typename _T2>
pair & std::pair< _T1, _T2 >::operator= (
            const pair< _T1, _T2 > & __p)  [inline], [constexpr], [noexcept]
```

Copy assignment operator.

operator=() [3/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_const_assignable<const _U1&, const _U2&>())
const pair & std::pair< _T1, _T2 >::operator= (
            const pair< _U1, _U2 > & __p) const  [inline], [constexpr]
```

Converting assignment from a const `pair<U1, U2>` lvalue.

operator=() [4/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_assignable<const _U1&, const _U2&>())
pair & std::pair< _T1, _T2 >::operator= (
            const pair< _U1, _U2 > & __p)  [inline], [constexpr], [noexcept]
```

Converting assignment from a const `pair<U1, U2>` lvalue.

operator=() [5/8]

```
template<typename _T1, typename _T2>
const pair & std::pair< _T1, _T2 >::operator= (
            pair< _T1, _T2 > && __p) const  [inline], [constexpr]
```

Move assignment operator (const).

operator=() [6/8]

```
template<typename _T1, typename _T2>
pair & std::pair< _T1, _T2 >::operator= (
            pair< _T1, _T2 > && __p)  [inline], [constexpr], [noexcept]
```

Move assignment operator.

operator=() [7/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_const_assignable<_U1, _U2>())
const pair & std::pair< _T1, _T2 >::operator= (
            pair< _U1, _U2 > && __p) const  [inline], [constexpr]
```

Converting assignment from a non-const `pair<U1, U2>` rvalue.

operator=() [8/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_assignable<_U1, _U2>())
pair & std::pair< _T1, _T2 >::operator= (
            pair< _U1, _U2 > && __p)  [inline], [constexpr], [noexcept]
```

Converting assignment from a non-const `pair<U1, U2>` rvalue.

swap()

```
template<typename _T1, typename _T2>
void std::pair< _T1, _T2 >::swap (
            pair< _T1, _T2 > & __p)  [inline], [constexpr], [noexcept]
```

Swap the first members and then the second members.

6.776.5 Member Data Documentation

first

```
template<typename _T1, typename _T2>
_T1 std::pair< _T1, _T2 >::first
```

The first member.

second

```
template<typename _T1, typename _T2>
_T2 std::pair< _T1, _T2 >::second
```

The second member.

The documentation for this struct was generated from the following files:

- bits/stl_iterator.h
- stl_pair.h
- tuple

6.777 std::pair< _T1, _T2 > Struct Template Reference

```
#include <utility>
```

Public Types

- typedef _T1 first_type
- typedef _T2 second_type

Public Member Functions

- constexpr pair (const pair &)=default
- constexpr pair (pair &&)=default
- template<typename... _Args1, typename... _Args2>
  constexpr pair (piecewise_construct_t, tuple< _Args1... >, tuple< _Args2... >)
- template<typename _U1, typename _U2>
  constexpr **explicit** (!_S_convertible< _U1 &, _U2 & >()) pair(pair< _U1
- template<typename _U1, typename _U2>
  constexpr _U2 &&constexpr explicit (!_S_convertible< _U1 &, _U2 & >()) pair(pair< _U1
- template<typename _U1 = _T1, typename _U2 = _T2>
  constexpr **explicit** (!_S_convertible< _U1, _U2 >()) pair(_U1 &&
- template<typename _U1 = _T1, typename _U2 = _T2>
  constexpr explicit (!_S_convertible< _U1, _U2 >()) pair(_U1 &&__x
- template<typename _U1, typename _U2>
  constexpr **explicit** (!_S_convertible< _U1, _U2 >()) pair(pair< _U1
- template<typename _U1, typename _U2>
  constexpr _U2 &constexpr explicit (!_S_convertible< _U1, _U2 >()) pair(pair< _U1
- constexpr explicit (!_S_convertible< const _T1 &, const _T2 & >()) pair(const type_identity_t< _T1 > &__x
- template<typename _U1, typename _U2>
  constexpr **explicit** (!_S_convertible< const _U1 &, const _U2 & >()) pair(const pair< _U1
- template<typename _U1, typename _U2>
  constexpr _U2 &&constexpr explicit (!_S_convertible< const _U1 &, const _U2 & >()) pair(const pair< _U1
- template<typename _U1, typename _U2>
  constexpr **explicit** (!_S_convertible< const _U1, const _U2 >()) pair(const pair< _U1
- template<typename _U1, typename _U2>
  constexpr _U2 &constexpr explicit (!_S_convertible< const _U1, const _U2 >()) pair(const pair< _U1
- template<__eligible_pair_like< pair > _UPair>
  constexpr **explicit** (!_S_convertible_from_pair_like< _UPair >()) pair(_UPair &&)=delete
- template<__eligible_pair_like< pair > _UPair>
  constexpr _U2 &&constexpr **explicit** (!_S_convertible_from_pair_like< _UPair >()) pair(_UPair &&__p)
- constexpr explicit (__not_< __and_< __is_implicitly_default_constructible< _T1 >, __is_implicitly_default_↩
  constructible< _T2 > > >()) pair() noexcept(is_nothrow_default_constructible_v< _T1 > &&is_nothrow_↩
  default_constructible_v< _T2 >)
- constexpr _U2 &&constexpr _U2 &__p **noexcept** (_S_nothrow_constructible< _U1 &, _U2 & >())
- constexpr _U2 &constexpr _U2 &&__p **noexcept** (_S_nothrow_constructible< _U1, _U2 >())
- constexpr _U2 &&__y **noexcept** (_S_nothrow_constructible< _U1, _U2 >())
- constexpr const _T2 &__y **noexcept** (_S_nothrow_constructible< const _T1 &, const _T2 & >())
- constexpr _U2 &&constexpr _U2 &__p **noexcept** (_S_nothrow_constructible< const _U1 &, const _U2 & >())
- constexpr _U2 &constexpr _U2 &&__p **noexcept** (_S_nothrow_constructible< const _U1, const _U2 >())
- template<__eligible_pair_like< pair > _UPair>
  constexpr pair & **operator=** (_UPair &&__p)
- template<__eligible_pair_like< pair > _UPair>
  constexpr const pair & **operator=** (_UPair &&__p) const
- pair & **operator=** (const pair &)=delete
- constexpr const pair & operator= (const pair &__p) const
- constexpr pair & operator= (const pair &__p) noexcept(_S_nothrow_assignable< const _T1 &, const _T2 & >())
- template<typename _U1, typename _U2>
  requires (_S_const_assignable<const _U1&, const _U2&>())
  constexpr const pair & operator= (const pair< _U1, _U2 > &__p) const

- template<typename _U1, typename _U2>
  requires (_S_assignable<const _U1&, const _U2&>())
  constexpr pair & operator= (const pair< _U1, _U2 > &__p) noexcept(_S_nothrow_assignable< const _U1 &, const _U2 & >())
- constexpr const pair & operator= (pair &&__p) const
- constexpr pair & operator= (pair &&__p) noexcept(_S_nothrow_assignable< _T1, _T2 >())
- template<typename _U1, typename _U2>
  requires (_S_const_assignable<_U1, _U2>())
  constexpr const pair & operator= (pair< _U1, _U2 > &&__p) const
- template<typename _U1, typename _U2>
  requires (_S_assignable<_U1, _U2>())
  constexpr pair & operator= (pair< _U1, _U2 > &&__p) noexcept(_S_nothrow_assignable< _U1, _U2 >())
- constexpr void **swap** (const pair &__p) const noexcept(__and_v< __is_nothrow_swappable< const _T1 >, _↩ _is_nothrow_swappable< const _T2 > >)
- constexpr void swap (pair &__p) noexcept(__and_< __is_nothrow_swappable< _T1 >, __is_nothrow_↩ swappable< _T2 > >::value)

Public Attributes

- _T1 first
- _T2 second

Related Symbols

(Note that these are not member symbols.)

- template<typename _T1, typename _T2>
  pair (_T1, _T2) -> pair< _T1, _T2 >
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr bool operator== (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)
- template<typename _T1, typename _T2, typename _U1, typename _U2>
  constexpr common_comparison_category_t< __detail::__synth3way_t< _T1, _U1 >, __detail::__synth3way_t< _T2, _U2 > > operator<=> (const pair< _T1, _T2 > &__x, const pair< _U1, _U2 > &__y)
- template<typename _T1, typename _T2>
  constexpr enable_if< __and_< __is_swappable< _T1 >, __is_swappable< _T2 > >::value >::type swap (pair< _T1, _T2 > &__x, pair< _T1, _T2 > &__y) noexcept(noexcept(__x.swap(__y)))

6.777.1 Detailed Description

template<typename _T1, typename _T2>
struct std::pair< _T1, _T2 >

Struct holding two objects (or references) of arbitrary type.

Template Parameters

_T1	Type of the `first` member.
_T2	Type of the `second` member.

https://gcc.gnu.org/onlinedocs/libstdc++/manual/utilities.html

6.777.2 Member Typedef Documentation

first_type

```
template<typename _T1, typename _T2>
typedef _T1 std::pair< _T1, _T2 >::first_type
```

The type of the `first` member.

second_type

```
template<typename _T1, typename _T2>
typedef _T2 std::pair< _T1, _T2 >::second_type
```

The type of the `second` member.

6.777.3 Constructor & Destructor Documentation

pair() [1/2]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::pair (
            const pair< _T1, _T2 > & )  [constexpr], [default]
```

Copy constructor.
Referenced by pair().

pair() [2/2]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::pair (
            pair< _T1, _T2 > && )  [constexpr], [default]
```

Move constructor.

6.777.4 Member Function Documentation

explicit() [1/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &&constexpr std::pair< _T1, _T2 >::explicit (
            !_S_convertible< _U1 &, _U2 & > ())  [constexpr]
```

Converting constructor from a non-const `pair<U1, U2>` lvalue.

explicit() [2/7]

```
template<typename _T1, typename _T2>
template<typename _U1 = _T1, typename _U2 = _T2>
std::pair< _T1, _T2 >::explicit (
            !_S_convertible< _U1, _U2 > ()) &&  [constexpr]
```

Constructor accepting two values of arbitrary types.

explicit() [3/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &constexpr std::pair< _T1, _T2 >::explicit (
            !_S_convertible< _U1, _U2 > ())  [constexpr]
```

Converting constructor from a non-const `pair<U1, U2>` rvalue.

explicit() [4/7]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::explicit (
            !_S_convertible< const _T1 &, const _T2 & > ()) const &  [constexpr]
```

Constructor accepting lvalues of `first_type` and `second_type`.

explicit() [5/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &&constexpr std::pair< _T1, _T2 >::explicit (
            !_S_convertible< const _U1 &, const _U2 & > ()) const  [constexpr]
```

Converting constructor from a const `pair<U1, U2>` lvalue.

explicit() [6/7]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
_U2 &constexpr std::pair< _T1, _T2 >::explicit (
            !_S_convertible< const _U1, const _U2 > ()) const  [constexpr]
```

Converting constructor from a const `pair<U1, U2>` rvalue.

explicit() [7/7]

```
template<typename _T1, typename _T2>
std::pair< _T1, _T2 >::explicit (
            __not_< __and_< __is_implicitly_default_constructible< _T1 >, __is_implicitly_↩
default_constructible< _T2 > > > ())  [inline], [constexpr], [noexcept]
```

Default constructor.

operator=() [1/8]

```
template<typename _T1, typename _T2>
const pair & std::pair< _T1, _T2 >::operator= (
            const pair< _T1, _T2 > & __p) const  [inline], [constexpr]
```

Copy assignment operator (const).

operator=() [2/8]

```
template<typename _T1, typename _T2>
pair & std::pair< _T1, _T2 >::operator= (
            const pair< _T1, _T2 > & __p)  [inline], [constexpr], [noexcept]
```

Copy assignment operator.

operator=() [3/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_const_assignable<const _U1&, const _U2&>())
const pair & std::pair< _T1, _T2 >::operator= (
            const pair< _U1, _U2 > & __p) const  [inline], [constexpr]
```

Converting assignment from a const `pair<U1, U2>` lvalue.

operator=() [4/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_assignable<const _U1&, const _U2&>())
pair & std::pair< _T1, _T2 >::operator= (
            const pair< _U1, _U2 > & __p)  [inline], [constexpr], [noexcept]
```

Converting assignment from a const pair<U1, U2> lvalue.

operator=() [5/8]

```
template<typename _T1, typename _T2>
const pair & std::pair< _T1, _T2 >::operator= (
            pair< _T1, _T2 > && __p) const  [inline], [constexpr]
```

Move assignment operator (const).

operator=() [6/8]

```
template<typename _T1, typename _T2>
pair & std::pair< _T1, _T2 >::operator= (
            pair< _T1, _T2 > && __p)  [inline], [constexpr], [noexcept]
```

Move assignment operator.

operator=() [7/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_const_assignable<_U1, _U2>())
const pair & std::pair< _T1, _T2 >::operator= (
            pair< _U1, _U2 > && __p) const  [inline], [constexpr]
```

Converting assignment from a non-const pair<U1, U2> rvalue.

operator=() [8/8]

```
template<typename _T1, typename _T2>
template<typename _U1, typename _U2>
requires (_S_assignable<_U1, _U2>())
pair & std::pair< _T1, _T2 >::operator= (
            pair< _U1, _U2 > && __p)  [inline], [constexpr], [noexcept]
```

Converting assignment from a non-const pair<U1, U2> rvalue.

swap()

```
template<typename _T1, typename _T2>
void std::pair< _T1, _T2 >::swap (
            pair< _T1, _T2 > & __p)  [inline], [constexpr], [noexcept]
```

Swap the first members and then the second members.
Referenced by std::pair< const __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator > >::swap(), std::sub_match< const char * and std::swap().

6.777.5 Member Data Documentation

first

```
template<typename _T1, typename _T2>
_T1 std::pair< _T1, _T2 >::first
```

The first member.
Referenced by __gnu_parallel::__find_template(), __gnu_parallel::__find_template(), std::set< _Key, _Cmp, polymorphic_allocator< _Key
std::pair< const __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator > >::operator<=>(), std::pair< const __iter_key_t< _Input
and std::shuffle().

second

```
template<typename _T1, typename _T2>
_T2 std::pair< _T1, _T2 >::second
```

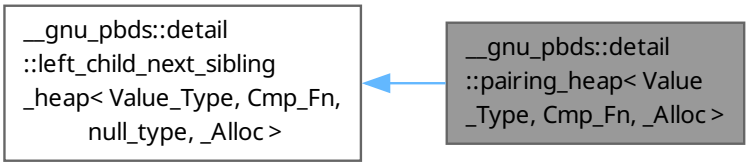
The second member.
Referenced by std::set< _Key, _Cmp, polymorphic_allocator< _Key > >::insert(), std::pair< const __iter_key_t< _InputIterator >, __iter
std::pair< const __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator > >::operator==(), and std::shuffle().
The documentation for this struct was generated from the following files:

- bits/stl_iterator.h
- stl_pair.h
- tuple

6.778 __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

```
#include <pairing_heap_.hpp>
```

Inheritance diagram for __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef __rebind_a::const_pointer **const_pointer**
- typedef __rebind_a::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef left_child_next_sibling_heap_node_< Value_Type, null_type, _Alloc > **node**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef __rebind_a::pointer **pointer**
- typedef __rebind_a::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **pairing_heap** (const Cmp_Fn &)
- **pairing_heap** (const pairing_heap &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- void **erase** (point_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **join** (pairing_heap &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred>
  void **split** (Pred, pairing_heap &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, null_type, _Alloc > &)
- void **swap** (pairing_heap &)
- const_reference **top** () const

Protected Types

- typedef alloc_traits::allocator_type **node_allocator**
- typedef alloc_traits::const_pointer **node_const_pointer**
- typedef null_type **node_metadata**
- typedef std::pair< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It>
  void **copy_from_range** (It, It)
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- size_type **m_size**

6.778.1 Detailed Description

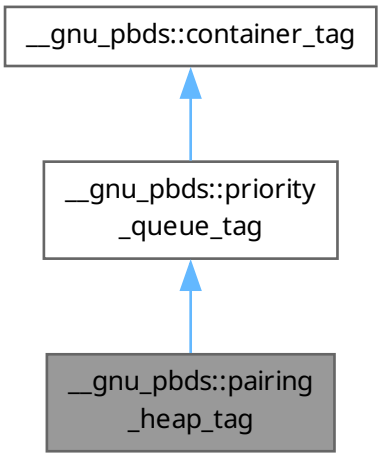
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >

Pairing heap.
The documentation for this class was generated from the following file:

- pairing_heap_.hpp

6.779 __gnu_pbds::pairing_heap_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::pairing_heap_tag:



6.779.1 Detailed Description

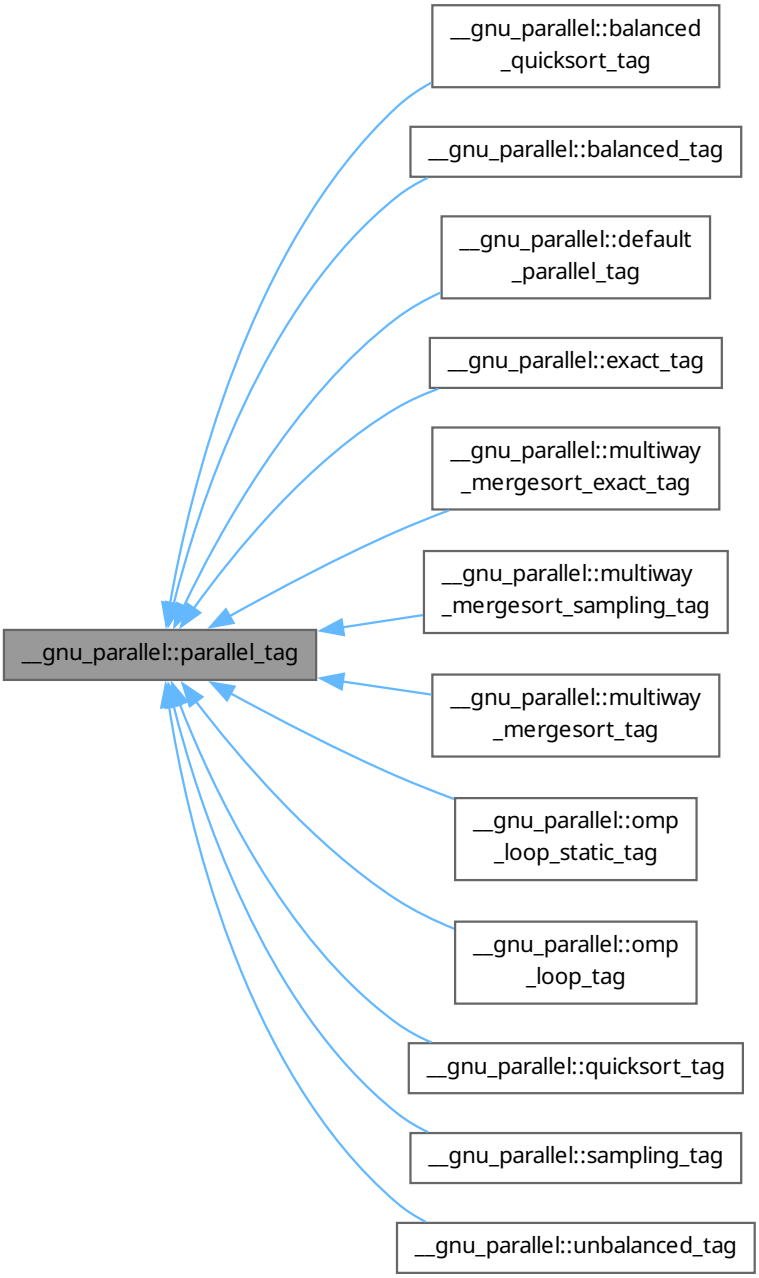
Pairing-heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.780 __gnu_parallel::parallel_tag Struct Reference

`#include <tags.h>`

Inheritance diagram for __gnu_parallel::parallel_tag:



Public Member Functions

- parallel_tag ()
- parallel_tag (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()

- void set_num_threads (_ThreadIndex __num_threads)

6.780.1 Detailed Description

Recommends parallel execution at compile time, optionally using a user-specified number of threads.

6.780.2 Constructor & Destructor Documentation

parallel_tag() [1/2]

```
__gnu_parallel::parallel_tag::parallel_tag ()  [inline]
```
Default constructor. Use default number of threads.

parallel_tag() [2/2]

```
__gnu_parallel::parallel_tag::parallel_tag (
            _ThreadIndex __num_threads)  [inline]
```
Default constructor. Recommend number of threads to use.

Parameters

__num_threads	Desired number of threads.

6.780.3 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline]
```
Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline]
```
Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

6.781 std::bernoulli_distribution::param_type Struct Reference

```
#include <random.h>
```

Public Types

- typedef bernoulli_distribution **distribution_type**

Public Member Functions

- **param_type** (double __p)
- double **p** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.781.1 Detailed Description

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

6.782 std::binomial_distribution< _IntType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef binomial_distribution< _IntType > **distribution_type**

Public Member Functions

- **param_type** (_IntType __t, double __p=0.5)
- double **p** () const
- _IntType **t** () const

Friends

- class **binomial_distribution**< **_IntType** >
- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.782.1 Detailed Description

template<**typename _IntType = int**>
struct std::binomial_distribution< **_IntType** >**::param_type**

Parameter type.
The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

6.783 std::cauchy_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef cauchy_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a, _RealType __b=_RealType(1))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.783.1 Detailed Description

template<typename _RealType = double>
struct std::cauchy_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

6.784 std::chi_squared_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef chi_squared_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __n)
- _RealType **n** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.784.1 Detailed Description

template<typename _RealType = double>
struct std::chi_squared_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

6.785 std::discrete_distribution< _IntType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef discrete_distribution< _IntType > **distribution_type**

Public Member Functions

- template<typename _InputIterator>
  param_type (_InputIterator __wbegin, _InputIterator __wend)
- **param_type** (const param_type &)=default
- **param_type** (initializer_list< double > __wil)
- template<typename _Func>
  param_type (size_t __nw, double __xmin, double __xmax, _Func __fw)
- param_type & **operator=** (const param_type &)=default
- std::vector< double > **probabilities** () const

Friends

- class **discrete_distribution**< **_IntType** >
- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.785.1 Detailed Description

template<typename _IntType = int>
struct std::discrete_distribution< _IntType >::param_type

Parameter type.
The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

6.786 std::exponential_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef exponential_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __lambda)
- _RealType **lambda** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.786.1 Detailed Description

template<typename _RealType = double>
struct std::exponential_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

6.787 std::extreme_value_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef extreme_value_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a, _RealType __b=_RealType(1.0))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.787.1 Detailed Description

template<typename _RealType = double>
struct std::extreme_value_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

6.788 std::fisher_f_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef fisher_f_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __m, _RealType __n=_RealType(1))
- _RealType **m** () const
- _RealType **n** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.788.1 Detailed Description

template<typename _RealType = double>
struct std::fisher_f_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

6.789 std::gamma_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef gamma_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __alpha_val, _RealType __beta_val=_RealType(1))
- _RealType **alpha** () const
- _RealType **beta** () const

Friends

- class **gamma_distribution**< **_RealType** >
- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.789.1 Detailed Description

template<**typename _RealType = double**>
struct std::gamma_distribution< **_RealType** >**::param_type**

Parameter type.
The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

6.790 std::geometric_distribution< _IntType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef geometric_distribution< _IntType > **distribution_type**

Public Member Functions

- **param_type** (double __p)
- double **p** () const

Friends

- class **geometric_distribution**< **_IntType** >
- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.790.1 Detailed Description

template<**typename _IntType = int**>
struct std::geometric_distribution< **_IntType** >**::param_type**

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

6.791 std::lognormal_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef lognormal_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __m, _RealType __s=_RealType(1))
- _RealType **m** () const
- _RealType **s** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.791.1 Detailed Description

template<typename _RealType = double>
struct std::lognormal_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

6.792 std::negative_binomial_distribution< _IntType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef negative_binomial_distribution< _IntType > **distribution_type**

Public Member Functions

- **param_type** (_IntType __k, double __p=0.5)
- _IntType **k** () const
- double **p** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.792.1 Detailed Description

template<typename _IntType = int>
struct std::negative_binomial_distribution< _IntType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

6.793 std::normal_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef normal_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __mean, _RealType __stddev=_RealType(1))
- _RealType **mean** () const
- _RealType **stddev** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.793.1 Detailed Description

template<typename _RealType = double>
struct std::normal_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

6.794 std::piecewise_constant_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef piecewise_constant_distribution< _RealType > **distribution_type**

Public Member Functions

- template<typename _InputIteratorB, typename _InputIteratorW>
  param_type (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- **param_type** (const param_type &)=default
- template<typename _Func>
  param_type (initializer_list< _RealType > __bi, _Func __fw)
- template<typename _Func>
  param_type (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- std::vector< result_type > **densities** () const
- std::vector< result_type > **intervals** () const
- param_type & **operator=** (const param_type &)=default

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator<<** (std::basic_ostream< _CharT, _Traits > &, const std::piecewise_constant_distribution< _RealType1 > &)
- bool **operator==** (const param_type &__p1, const param_type &__p2)
- class **piecewise_constant_distribution< _RealType >**

6.794.1 Detailed Description

template<typename _RealType = double>
struct std::piecewise_constant_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

6.795 std::piecewise_linear_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef piecewise_linear_distribution< _RealType > **distribution_type**

Public Member Functions

- template<typename _InputIteratorB, typename _InputIteratorW>
  param_type (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- **param_type** (const param_type &)=default
- template<typename _Func>
  param_type (initializer_list< _RealType > __bl, _Func __fw)
- template<typename _Func>
  param_type (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- std::vector< result_type > **densities** () const
- std::vector< result_type > **intervals** () const
- param_type & **operator=** (const param_type &)=default

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **operator<<** (std::basic_ostream< _CharT, _Traits > &, const std::piecewise_linear_distribution< _RealType1 > &)
- bool **operator==** (const param_type &__p1, const param_type &__p2)
- class **piecewise_linear_distribution< _RealType >**

6.795.1 Detailed Description

template<typename _RealType = double>
struct std::piecewise_linear_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

6.796 std::poisson_distribution< _IntType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef poisson_distribution< _IntType > **distribution_type**

Public Member Functions

- **param_type** (double __mean)
- double **mean** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)
- class **poisson_distribution< _IntType >**

6.796.1 Detailed Description

template<typename _IntType = int>
struct std::poisson_distribution< _IntType >::param_type

Parameter type.
The documentation for this struct was generated from the following files:

- random.h
- bits/random.tcc

6.797 std::student_t_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef student_t_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __n)
- _RealType **n** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.797.1 Detailed Description

template<typename _RealType = double>
struct std::student_t_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

6.798 std::uniform_int_distribution< _IntType >::param_type Struct Reference

`#include <uniform_int_dist.h>`

Public Types

- typedef uniform_int_distribution< _IntType > **distribution_type**

Public Member Functions

- **param_type** (_IntType __a, _IntType __b=__gnu_cxx::__int_traits< _IntType >::__max)
- result_type **a** () const
- result_type **b** () const

Friends

- bool **operator!=** (const param_type &__p1, const param_type &__p2)
- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.798.1 Detailed Description

template<typename _IntType = int>
struct std::uniform_int_distribution< _IntType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- uniform_int_dist.h

6.799 std::uniform_real_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef uniform_real_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a, _RealType __b=_RealType(1))
- result_type **a** () const
- result_type **b** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.799.1 Detailed Description

template<typename _RealType = double>
struct std::uniform_real_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

6.800 std::weibull_distribution< _RealType >::param_type Struct Reference

`#include <random.h>`

Public Types

- typedef weibull_distribution< _RealType > **distribution_type**

Public Member Functions

- **param_type** (_RealType __a, _RealType __b=_RealType(1.0))
- _RealType **a** () const
- _RealType **b** () const

Friends

- bool **operator==** (const param_type &__p1, const param_type &__p2)

6.800.1 Detailed Description

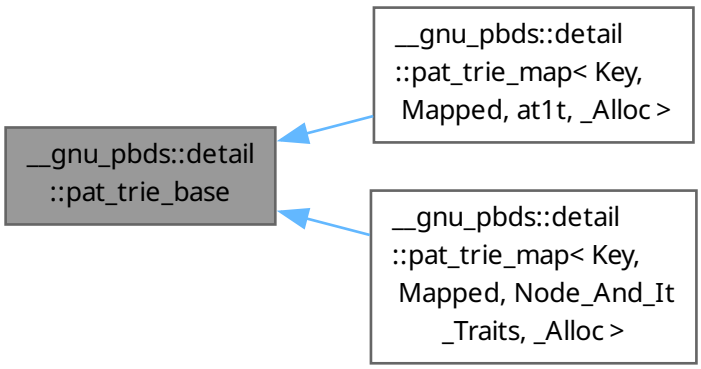
template<typename _RealType = double>
struct std::weibull_distribution< _RealType >::param_type

Parameter type.
The documentation for this struct was generated from the following file:

- random.h

6.801 __gnu_pbds::detail::pat_trie_base Struct Reference

`#include <pat_trie_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_base:



Classes

- class _CIter
- struct _Head
- struct _Inode
- class _Iter
- struct _Leaf

- struct _Metadata
- struct _Metadata< null_type, _Alloc >
- struct _Node_base
- class _Node_citer
- class _Node_iter

Public Types

- enum node_type { **i_node** , **leaf_node** , **head_node** }

6.801.1 Detailed Description

Base type for PATRICIA trees.

6.801.2 Member Enumeration Documentation

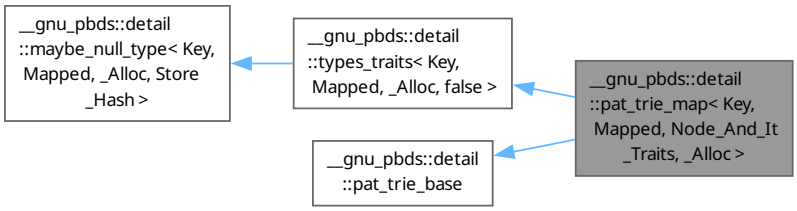
node_type

`enum __gnu_pbds::detail::pat_trie_base::node_type`
Three types of nodes.
i_node is used by _Inode, leaf_node by _Leaf, and head_node by _Head.
The documentation for this struct was generated from the following file:

- pat_trie_base.hpp

6.802 __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > Class Template Reference

`#include <pat_trie_.hpp>`
Inheritance diagram for __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >:



Public Types

- typedef traits_type::access_traits **access_traits**
- typedef _Alloc **allocator_type**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef point_const_iterator **const_iterator**
- typedef traits_base::const_pointer **const_pointer**
- typedef traits_base::const_reference **const_reference**
- typedef traits_type::const_reverse_iterator **const_reverse_iterator**
- typedef pat_trie_tag **container_category**
- typedef _Alloc::difference_type **difference_type**

- typedef point_iterator **iterator**
- typedef traits_base::key_const_pointer **key_const_pointer**
- typedef traits_base::key_const_reference **key_const_reference**
- typedef traits_base::key_pointer **key_pointer**
- typedef traits_base::key_reference **key_reference**
- typedef traits_base::key_type **key_type**
- typedef traits_base::mapped_const_pointer **mapped_const_pointer**
- typedef traits_base::mapped_const_reference **mapped_const_reference**
- typedef traits_base::mapped_pointer **mapped_pointer**
- typedef traits_base::mapped_reference **mapped_reference**
- typedef traits_base::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- enum node_type { **i_node** , **leaf_node** , **head_node** }
- typedef traits_type::node_update **node_update**
- typedef traits_type::const_iterator **point_const_iterator**
- typedef traits_type::iterator **point_iterator**
- typedef traits_base::pointer **pointer**
- typedef traits_base::reference **reference**
- typedef traits_type::reverse_iterator **reverse_iterator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef traits_base::value_type **value_type**

Public Member Functions

- **pat_trie_map** (const access_traits &)
- **pat_trie_map** (const pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- const_iterator **erase** (const_iterator)
- const_reverse_iterator **erase** (const_reverse_iterator)
- iterator **erase** (iterator)
- bool **erase** (key_const_reference)
- reverse_iterator **erase** (reverse_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- access_traits & **get_access_traits** ()
- const access_traits & **get_access_traits** () const
- node_update & **get_node_update** ()
- const node_update & **get_node_update** () const
- std::pair< point_iterator, bool > **insert** (const_reference)
- void **join** (pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)

- point_iterator **lower_bound** (key_const_reference)
- point_const_iterator **lower_bound** (key_const_reference) const
- size_type **max_size** () const
- node_iterator node_begin ()
- node_const_iterator node_begin () const
- node_iterator node_end ()
- node_const_iterator node_end () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- size_type **size** () const
- void **split** (key_const_reference, pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- void **swap** (pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference)
- point_const_iterator **upper_bound** (key_const_reference) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Protected Member Functions

- template<typename It>
  void **copy_from_range** (It, It)
- node_pointer **recursive_copy_node** (node_const_pointer)
- void **value_swap** (pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc > &)

6.802.1 Detailed Description

template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >

PATRICIA trie.
This implementation loosely borrows ideas from: 1) Fast Mergeable Integer Maps, Okasaki, Gill 1998 2) Ptset: Sets of integers implemented as Patricia trees, Jean-Christophe Filliatr, 2000

6.802.2 Member Enumeration Documentation

node_type

```
enum __gnu_pbds::detail::pat_trie_base::node_type  [inherited]
```

Three types of nodes.
i_node is used by _Inode, leaf_node by _Leaf, and head_node by _Head.

6.802.3 Member Function Documentation

node_begin() [1/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin ()  [inline]
```

Returns a node_iterator corresponding to the node at the root of the tree.

node_begin() [2/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_begin () const  [inline]
```

Returns a const node_iterator corresponding to the node at the root of the tree.

node_end() [1/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_end ()  [inline]
```

Returns a node_iterator corresponding to a node just after a leaf of the tree.

node_end() [2/2]

```
template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::pat_trie_map<
Key, Mapped, Node_And_It_Traits, _Alloc >::node_end () const  [inline]
```

Returns a const node_iterator corresponding to a node just after a leaf of the tree.

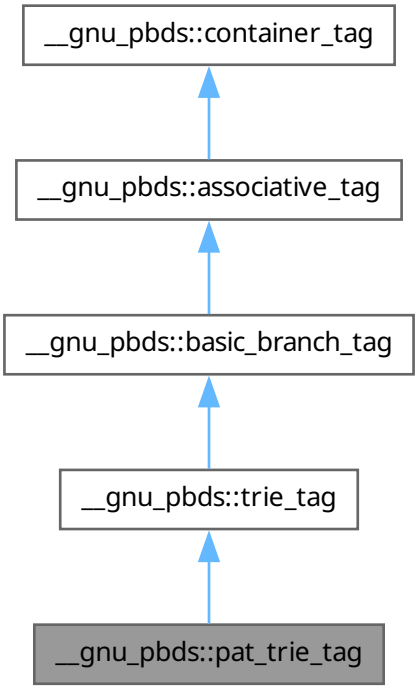
The documentation for this class was generated from the following file:

- pat_trie_.hpp

6.803 __gnu_pbds::pat_trie_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::pat_trie_tag:



6.803.1 Detailed Description

PATRICIA trie.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.804 std::experimental::filesystem::v1::path Class Reference

`#include <fs_path.h>`

Classes

- class iterator

Public Types

- typedef iterator **const_iterator**
- typedef std::basic_string< value_type > **string_type**
- typedef char **value_type**

Public Member Functions

- template<typename _InputIterator, typename _Require = __detail::_Path<_InputIterator, _InputIterator>>
  path (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator, typename _Require = __detail::_Path<_InputIterator, _InputIterator>, typename _Require2 = __detail::__value_type_is_char<_InputIterator>>
  path (_InputIterator __first, _InputIterator __last, const locale &__loc)
- template<typename _Source, typename _Require = __detail::_Path<_Source>>
  path (_Source const &__source)
- template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _Require2 = __detail::__value_type_is_char<_Source>>
  path (_Source const &__source, const locale &__loc)
- **path** (const path &__p)
- **path** (path &&__p) noexcept
- **path** (string_type &&__source)
- template<typename _InputIterator>
  __detail::_Path< _InputIterator, _InputIterator > & **append** (_InputIterator __first, _InputIterator __last)
- template<typename _Source>
  __detail::_Path< _Source > & **append** (_Source const &__source)
- template<typename _InputIterator>
  __detail::_Path< _InputIterator, _InputIterator > & **assign** (_InputIterator __first, _InputIterator __last)
- template<typename _Source>
  __detail::_Path< _Source > & **assign** (_Source const &__source)
- path & **assign** (string_type &&__source)
- iterator **begin** () const noexcept
- const value_type * **c_str** () const noexcept
- void **clear** () noexcept
- int **compare** (const basic_string_view< value_type > __s) const
- int **compare** (const path &__p) const noexcept
- int **compare** (const string_type &__s) const
- int **compare** (const value_type *__s) const
- template<typename _InputIterator>
  __detail::_Path< _InputIterator, _InputIterator > & **concat** (_InputIterator __first, _InputIterator __last)
- template<typename _Source>
  __detail::_Path< _Source > & **concat** (_Source const &__x)
- bool **empty** () const noexcept
- iterator **end** () const noexcept
- path **extension** () const
- path **filename** () const
- std::string **generic_string** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
  std::basic_string< _CharT, _Traits, _Allocator > **generic_string** (const _Allocator &__a=_Allocator()) const
- std::u16string **generic_u16string** () const
- std::u32string **generic_u32string** () const
- std::string **generic_u8string** () const
- std::wstring **generic_wstring** () const
- bool **has_extension** () const
- bool **has_filename** () const
- bool **has_parent_path** () const
- bool **has_relative_path** () const
- bool **has_root_directory** () const
- bool **has_root_name** () const

- bool **has_root_path** () const
- bool **has_stem** () const
- bool **is_absolute** () const
- bool **is_relative** () const
- path & **make_preferred** ()
- const string_type & **native** () const noexcept
- **operator string_type** () const
- template<typename _CharT>
  __detail::_Path< _CharT *, _CharT * > & **operator+=** (_CharT __x)
- template<typename _Source>
  __detail::_Path< _Source > & **operator+=** (_Source const &__x)
- path & **operator+=** (basic_string_view< value_type > __x)
- path & **operator+=** (const path &__x)
- path & **operator+=** (const string_type &__x)
- path & **operator+=** (const value_type *__x)
- path & **operator+=** (value_type __x)
- template<typename _Source>
  __detail::_Path< _Source > & **operator/=** (_Source const &__source)
- path & **operator/=** (const path &__p)
- template<typename _Source>
  __detail::_Path< _Source > & **operator=** (_Source const &__source)
- path & **operator=** (const path &__p)
- path & **operator=** (path &&__p) noexcept
- path & **operator=** (string_type &&__source)
- path **parent_path** () const
- path **relative_path** () const
- path & **remove_filename** ()
- path & **replace_extension** (const path &__replacement=path())
- path & **replace_filename** (const path &__replacement)
- path **root_directory** () const
- path **root_name** () const
- path **root_path** () const
- path **stem** () const
- std::string **string** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
  std::basic_string< _CharT, _Traits, _Allocator > **string** (const _Allocator &__a=_Allocator()) const
- void **swap** (path &__rhs) noexcept
- std::u16string **u16string** () const
- std::u32string **u32string** () const
- std::string **u8string** () const
- std::wstring **wstring** () const

Static Public Attributes

- static constexpr value_type **preferred_separator**

6.804.1 Detailed Description

A filesystem path.
The documentation for this class was generated from the following file:

- experimental/bits/fs_path.h

6.805 std::filesystem::path Class Reference

`#include <filesystem>`

Classes

- class iterator

Public Types

- using **const_iterator**
- enum format : unsigned char { **native_format** , **generic_format** , **auto_format** }
- using **string_type**
- using **value_type**

Public Member Functions

- template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename _Req2 = __detail::__value_type_is_char<_InputIterator>>
  path (_InputIterator __first, _InputIterator __last, const locale &__loc, format=auto_format)
- template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>>
  path (_InputIterator __first, _InputIterator __last, format=auto_format)
- template<typename _Source, typename _Require = __detail::_Path<_Source>>
  path (_Source const &__source, format=auto_format)
- template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _Require2 = __detail::__value_type_is_char<_Source>>
  path (_Source const &__src, const locale &__loc, format=auto_format)
- **path** (const path &__p)=default
- **path** (path &&__p) noexcept
- **path** (string_type &&__source, format=auto_format)
- template<typename _InputIterator>
  __detail::_Path2< _InputIterator > & **append** (_InputIterator __first, _InputIterator __last)
- template<typename _Source>
  __detail::_Path< _Source > & **append** (_Source const &__source)
- template<typename _InputIterator>
  __detail::_Path2< _InputIterator > & **assign** (_InputIterator __first, _InputIterator __last)
- template<typename _Source>
  __detail::_Path< _Source > & **assign** (_Source const &__source)
- path & **assign** (string_type &&__source)
- iterator **begin** () const noexcept
- const value_type * **c_str** () const noexcept
- void **clear** () noexcept
- int **compare** (basic_string_view< value_type > __s) const noexcept
- int **compare** (const path &__p) const noexcept
- int **compare** (const string_type &__s) const noexcept
- int **compare** (const value_type *__s) const noexcept
- template<typename _InputIterator>
  __detail::_Path2< _InputIterator > & **concat** (_InputIterator __first, _InputIterator __last)
- template<typename _Source>
  __detail::_Path< _Source > & **concat** (_Source const &__x)
- bool **empty** () const noexcept
- iterator **end** () const noexcept
- path **extension** () const

- path **filename** () const
- std::string **generic_string** () const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
  std::basic_string< _CharT, _Traits, _Allocator > **generic_string** (const _Allocator &__a=_Allocator()) const
- std::u16string **generic_u16string** () const
- std::u32string **generic_u32string** () const
- std::string **generic_u8string** () const
- std::wstring **generic_wstring** () const
- bool **has_extension** () const noexcept
- bool **has_filename** () const noexcept
- bool **has_parent_path** () const noexcept
- bool **has_relative_path** () const noexcept
- bool **has_root_directory** () const noexcept
- bool **has_root_name** () const noexcept
- bool **has_root_path** () const noexcept
- bool **has_stem** () const noexcept
- bool **is_absolute** () const noexcept
- bool **is_relative** () const noexcept
- path **lexically_normal** () const
- path **lexically_proximate** (const path &base) const
- path **lexically_relative** (const path &base) const
- path & **make_preferred** ()
- const string_type & **native** () const noexcept
- **operator string_type** () const
- template<typename _CharT>
  __detail::_Path2< _CharT * > & **operator+=** (_CharT __x)
- template<typename _Source>
  __detail::_Path< _Source > & **operator+=** (_Source const &__x)
- path & **operator+=** (basic_string_view< value_type > __x)
- path & **operator+=** (const path &__x)
- path & **operator+=** (const string_type &__x)
- path & **operator+=** (const value_type *__x)
- path & **operator+=** (value_type __x)
- template<typename _Source>
  __detail::_Path< _Source > & **operator/=** (_Source const &__source)
- path & **operator/=** (const path &__p)
- template<typename _Source>
  __detail::_Path< _Source > & **operator=** (_Source const &__source)
- path & **operator=** (const path &)
- path & **operator=** (path &&) noexcept
- path & **operator=** (string_type &&__source)
- path **parent_path** () const
- path **relative_path** () const
- path & **remove_filename** ()
- path & **replace_extension** (const path &__replacement=path())
- path & **replace_filename** (const path &__replacement)
- path **root_directory** () const
- path **root_name** () const
- path **root_path** () const
- path **stem** () const
- std::string **string** () const

- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **string** (const _Allocator &__a) const
- template<typename _CharT, typename _Traits = std::char_traits<_CharT>, typename _Allocator = std::allocator<_CharT>>
  std::basic_string< _CharT, _Traits, _Allocator > **string** (const _Allocator &__a=_Allocator()) const
- void **swap** (path &__rhs) noexcept
- std::u16string **u16string** () const
- std::u32string **u32string** () const
- std::string **u8string** () const
- std::wstring **wstring** () const

Static Public Attributes

- static constexpr value_type **preferred_separator**

Friends

- path operator/ (const path &__lhs, const path &__rhs)
- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const path &__p)
- strong_ordering operator<=> (const path &__lhs, const path &__rhs) noexcept
- bool operator== (const path &__lhs, const path &__rhs) noexcept
- template<typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, path &__p)

Related Symbols

(Note that these are not member symbols.)

- void **swap** (path &__lhs, path &__rhs) noexcept
- template<typename _InputIterator, typename _Require = __detail::_Path2<_InputIterator>, typename _CharT = __detail::__value_type_↩
  is_char_or_char8_t<_InputIterator>>
  path u8path (_InputIterator __first, _InputIterator __last)
- template<typename _Source, typename _Require = __detail::_Path<_Source>, typename _CharT = __detail::__value_type_is_char_or↩
  _char8_t<_Source>>
  path u8path (const _Source &__source)

6.805.1 Detailed Description

A filesystem path.

Since

C++17

6.805.2 Member Enumeration Documentation

format

```
enum std::filesystem::path::format :  unsigned char
```

path::format is ignored in this implementation

6.805.3 Friends And Related Symbol Documentation

operator/

```
path operator/ (
            const path & __lhs,
            const path & __rhs)  [friend]
```

Append one path to another.

operator<<

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const path & __p)  [friend]
```

Write a path to a stream.

operator<=>

```
strong_ordering operator<=> (
            const path & __lhs,
            const path & __rhs)  [friend]
```

Compare paths.

operator==

```
bool operator== (
            const path & __lhs,
            const path & __rhs)  [friend]
```

Compare paths.

operator>>

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            path & __p)  [friend]
```

Read a path from a stream.
References std::move().
The documentation for this class was generated from the following file:

- bits/fs_path.h

6.806 std::philox_engine< _UIntType, __w, __n, __r, __consts > Class Template Reference

`#include <random>`

Public Types

- using **result_type**

Public Member Functions

- template<typename _Sseq>
  philox_engine (_Sseq &__q)
- **philox_engine** (result_type __value)

- void discard (unsigned long long __z)
- result_type operator() ()
- template<typename _Sseq>
  requires __is_seed_seq<_Sseq>
  void seed (_Sseq &__q)
- void **seed** (result_type __value=default_seed)
- void set_counter (const array< result_type, __n > &__counter)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr result_type **default_seed**
- static constexpr array< result_type, __n/2 > **multipliers**
- static constexpr array< result_type, __n/2 > **round_consts**
- static constexpr size_t **round_count**
- static constexpr size_t **word_count**
- static constexpr size_t **word_size**

Friends

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const philox_engine &__x)
- bool operator== (const philox_engine &, const philox_engine &)=default
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & operator>> (basic_istream< _CharT, _Traits > &__is, philox_engine &__x)

6.806.1 Detailed Description

template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType... __consts>
class std::philox_engine< _UIntType, __w, __n, __r, __consts >

A discrete pseudorandom number generator with weak cryptographic properties.
This algorithm was designed to be used for highly parallel random number generation, and is capable of immensely long periods. It provides "Crush-resistance", denoting an ability to pass the TestU01 Suite's "Big Crush" test, demonstrating significant apparent entropy.
It is not intended for cryptographic use and should not be used for such, despite being based on cryptographic primitives. The typedefs `philox4x32` and `philox4x64` are provided as suitable defaults for most use cases, providing high-quality random numbers with reasonable performance.
This algorithm was created by John Salmon, Mark Moraes, Ron Dror, and David Shaw as a product of D.E. Shaw Research.

Template Parameters

__w	Word size
__n	Buffer size

__r	Rounds
__consts	Multiplication and round constant pack, ordered as M_{0}, C_{0}, M_{1}, C_{1}, ... , M_{N/2-1}, C_{N/2-1}

Since

C++26

6.806.2 Constructor & Destructor Documentation

philox_engine()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType...  __consts>
template<typename _Sseq>
std::philox_engine< _UIntType, __w, __n, __r, __consts >::philox_engine (
            _Sseq & __q)  [inline], [explicit]
```

seed sequence constructor for philox_engine

Parameters

_↩ _q	the seed sequence

6.806.3 Member Function Documentation

discard()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType...  __consts>
void std::philox_engine< _UIntType, __w, __n, __r, __consts >::discard (
            unsigned long long __z)  [inline]
```

discards __z numbers

Parameters

_↩ _z	number of iterations to discard

max()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType...  __consts>
constexpr result_type std::philox_engine< _UIntType, __w, __n, __r, __consts >::max ()  [inline],
[static], [constexpr]
```

The maximum value that this engine can return.

Referenced by std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::operator>>, and std::philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 >::set_↩ counter().

min()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType...  __consts>
constexpr result_type std::philox_engine< _UIntType, __w, __n, __r, __consts >::min ()  [inline],
[static], [constexpr]
```

The minimum value that this engine can return.

operator()()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType...  __consts>
result_type std::philox_engine< _UIntType, __w, __n, __r, __consts >::operator() ()  [inline]
```

outputs a single w-bit number and handles state advancement

Returns

return_type

seed()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType...  __consts>
requires __is_seed_seq<_Sseq>
template<typename _Sseq>
requires __is_seed_seq<_Sseq>
void std::philox_engine< _UIntType, __w, __n, __r, __consts >::seed (
            _Sseq & __q)
```

seeds philox_engine by seed sequence

Parameters

_↩ _q	the seed sequence

References std::max().

set_counter()

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType...  __consts>
void std::philox_engine< _UIntType, __w, __n, __r, __consts >::set_counter (
            const array< result_type, __n > & __counter)  [inline]
```

sets the internal counter "cleartext"

Parameters

__counter	std::array of len N

6.806.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType...  __consts>
template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const philox_engine< _UIntType, __w, __n, __r, __consts > & __x)  [friend]
```

outputs the state of the generator

Parameters

__os	An output stream.
__x	A philox_engine object reference

Returns

the state of the Philox Engine in __os

operator==

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType...  __consts>
bool operator== (
             const philox_engine< _UIntType, __w, __n, __r, __consts > & ,
             const philox_engine< _UIntType, __w, __n, __r, __consts > & )  [friend]
```

compares two philox_engine objects

Returns

true if the objects will produce an identical stream, false otherwise

operator>>

```
template<typename _UIntType, size_t __w, size_t __n, size_t __r, _UIntType...  __consts>
template<typename _CharT, typename _Traits>
basic_istream< _CharT, _Traits > & operator>> (
             basic_istream< _CharT, _Traits > & __is,
             philox_engine< _UIntType, __w, __n, __r, __consts > & __x)  [friend]
```

takes input to set the state of the philox_engine object

Parameters

_↩ _is	An input stream.
_↩ _x	A philox_engine object reference

Returns

philox_engine object is set with values from instream

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.807 std::piecewise_constant_distribution< _RealType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- template<typename _InputIteratorB, typename _InputIteratorW>
  piecewise_constant_distribution (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- **piecewise_constant_distribution** (const param_type &__p)
- template<typename _Func>
  piecewise_constant_distribution (initializer_list< _RealType > __bl, _Func __fw)
- template<typename _Func>
  piecewise_constant_distribution (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- std::vector< result_type > densities () const
- std::vector< result_type > intervals () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::piecewise_constant_distribution< _RealType1 > &__x)
- bool operator== (const piecewise_constant_distribution &__d1, const piecewise_constant_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::piecewise_constant_distribution< _RealType1 > &__x)

6.807.1 Detailed Description

template<typename _RealType = double>
class std::piecewise_constant_distribution< _RealType >

A piecewise_constant_distribution random number distribution.

This distribution produces random numbers $x, b_0 \leq x < b_n$, uniformly distributed over each subinterval $[b_i, b_{i+1})$ according to the probability mass function

$$p(x|b_0, ..., b_n, \rho_0, ..., \rho_{n-1}) = \rho_i \cdot \frac{b_{i+1} - x}{b_{i+1} - b_i} + \rho_{i+1} \cdot \frac{x - b_i}{b_{i+1} - b_i}$$

for $b_i \leq x < b_{i+1}$.

Since

C++11

6.807.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::piecewise_constant_distribution< _RealType >::result_type
```

The type of the range of the distribution.

6.807.3 Member Function Documentation

densities()

```
template<typename _RealType = double>
std::vector< result_type > std::piecewise_constant_distribution< _RealType >::densities () const
[inline]
```

Returns a vector of the probability densities.

intervals()

```
template<typename _RealType = double>
std::vector< result_type > std::piecewise_constant_distribution< _RealType >::intervals () const
[inline]
```

Returns a vector of the intervals.

max()

```
template<typename _RealType = double>
result_type std::piecewise_constant_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::piecewise_constant_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::piecewise_constant_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.

Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::piecewise_constant_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::piecewise_constant_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::piecewise_constant_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

6.807.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::piecewise_constant_distribution< _RealType1 > & __x)  [friend]
```

Inserts a piecewise_constant_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A piecewise_constant_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
            const piecewise_constant_distribution< _RealType > & __d1,
            const piecewise_constant_distribution< _RealType > & __d2)  [friend]
```

Return true if two piecewise constant distributions have the same parameters.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::piecewise_constant_distribution< _RealType1 > & __x)  [friend]
```

Extracts a piecewise_constant_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A piecewise_constant_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.808 std::piecewise_construct_t Struct Reference

```
#include <stl_pair.h>
```

6.808.1 Detailed Description

Tag type for piecewise construction of std::pair objects.
The documentation for this struct was generated from the following file:

- stl_pair.h

6.809 std::piecewise_linear_distribution< _RealType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- template<typename _InputIteratorB, typename _InputIteratorW>
  piecewise_linear_distribution (_InputIteratorB __bfirst, _InputIteratorB __bend, _InputIteratorW __wbegin)
- **piecewise_linear_distribution** (const param_type &__p)
- template<typename _Func>
  piecewise_linear_distribution (initializer_list< _RealType > __bl, _Func __fw)

- template<typename _Func>
  piecewise_linear_distribution (size_t __nw, _RealType __xmin, _RealType __xmax, _Func __fw)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- std::vector< result_type > densities () const
- std::vector< result_type > intervals () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::piecewise_linear_distribution< _RealType1 > &__x)
- bool operator== (const piecewise_linear_distribution &__d1, const piecewise_linear_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::piecewise_linear_distribution< _RealType1 > &__x)

6.809.1 Detailed Description

template<typename _RealType = double>
class std::piecewise_linear_distribution< _RealType >

A piecewise_linear_distribution random number distribution.
This distribution produces random numbers $x, b_0 \leq x < b_n$, distributed over each subinterval $[b_i, b_{i+1})$ according to the probability mass function $p(x|b_0, ..., b_n, \rho_0, ..., \rho_n) = \rho_i$, for $b_i \leq x < b_{i+1}$.

Since

C++11

6.809.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::piecewise_linear_distribution< _RealType >::result_type
```

The type of the range of the distribution.

6.809.3 Member Function Documentation

densities()

```
template<typename _RealType = double>
std::vector< result_type > std::piecewise_linear_distribution< _RealType >::densities () const
[inline]
```

Return a vector of the probability densities of the distribution.

intervals()

```
template<typename _RealType = double>
std::vector< result_type > std::piecewise_linear_distribution< _RealType >::intervals () const
[inline]
```

Return the intervals of the distribution.

max()

```
template<typename _RealType = double>
result_type std::piecewise_linear_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::piecewise_linear_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::piecewise_linear_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::piecewise_linear_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::piecewise_linear_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::piecewise_linear_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

6.809.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::piecewise_linear_distribution< _RealType1 > & __x)  [friend]
```

Inserts a piecewise_linear_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A piecewise_linear_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
            const piecewise_linear_distribution< _RealType > & __d1,
            const piecewise_linear_distribution< _RealType > & __d2)  [friend]
```

Return true if two piecewise linear distributions have the same parameters.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::piecewise_linear_distribution< _RealType1 > & __x)  [friend]
```

Extracts a piecewise_linear_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A piecewise_linear_distribution random number generator engine.

Returns

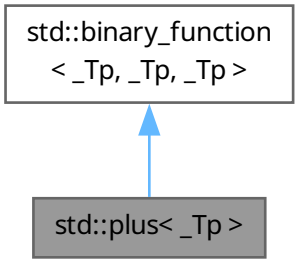
The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.810 std::plus< _Tp > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::plus< _Tp >:



Public Types

- typedef _Tp first_argument_type
- typedef _Tp result_type
- typedef _Tp second_argument_type

Public Member Functions

- constexpr _Tp operator() (const _Tp &__x, const _Tp &__y) const

6.810.1 Detailed Description

template<typename _Tp>
struct std::plus< _Tp >

One of the math functors.

6.810.2 Member Typedef Documentation

first_argument_type

`typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::first_argument_type [inherited]`
`first_argument_type` is the type of the first argument

result_type

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::result_type  [inherited]
```
result_type is the return type

second_argument_type

```
typedef _Tp std::binary_function< _Tp, _Tp, _Tp >::second_argument_type  [inherited]
```
second_argument_type is the type of the second argument

6.810.3 Member Function Documentation

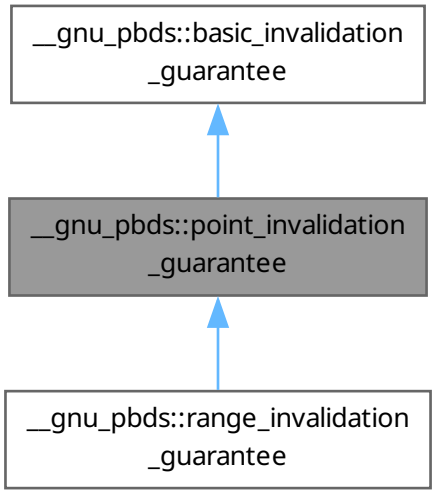
operator()()

```
template<typename _Tp>
_Tp std::plus< _Tp >::operator() (
            const _Tp & __x,
            const _Tp & __y) const  [inline], [constexpr]
```
Returns the sum.
The documentation for this struct was generated from the following file:

- stl_function.h

6.811 __gnu_pbds::point_invalidation_guarantee Struct Reference

```
#include <tag_and_trait.hpp>
```
Inheritance diagram for __gnu_pbds::point_invalidation_guarantee:



6.811.1 Detailed Description

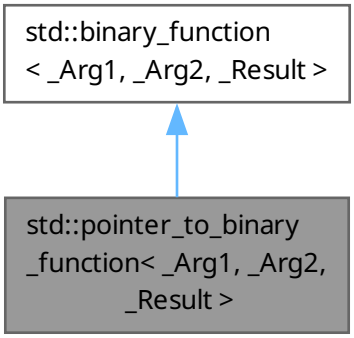
Signifies an invalidation guarantee that includes all those of its base, and additionally, that any point-type iterator, pointer, or reference to a container object's mapped value type is valid as long as its corresponding entry has not be erased,

regardless of modifications to the container object.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.812 std::pointer_to_binary_function< _Arg1, _Arg2, _Result > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::pointer_to_binary_function< _Arg1, _Arg2, _Result >:



Public Types

- typedef _Arg1 first_argument_type
- typedef _Result result_type
- typedef _Arg2 second_argument_type

Public Member Functions

- **pointer_to_binary_function** (_Result(*__x)(_Arg1, _Arg2))
- _Result **operator()** (_Arg1 __x, _Arg2 __y) const

Protected Attributes

- _Result(* **_M_ptr**)(_Arg1, _Arg2)

6.812.1 Detailed Description

template<typename _Arg1, typename _Arg2, typename _Result>
class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >

One of the adaptors for function pointers.

6.812.2 Member Typedef Documentation

first_argument_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Result >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Result std::binary_function< _Arg1, _Arg2, _Result >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

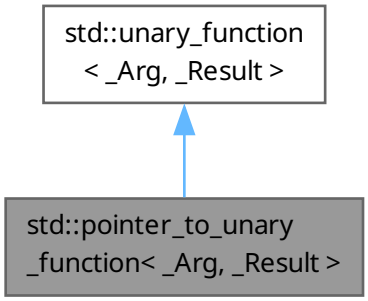
```
template<typename _Arg1, typename _Arg2, typename _Result>
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Result >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this class was generated from the following file:

- stl_function.h

6.813 std::pointer_to_unary_function< _Arg, _Result > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::pointer_to_unary_function< _Arg, _Result >:



Public Types

- typedef _Arg argument_type
- typedef _Result result_type

Public Member Functions

- **pointer_to_unary_function** (_Result(*__x)(_Arg))
- _Result **operator()** (_Arg __x) const

Protected Attributes

- _Result(∗ **_M_ptr**)(_Arg)

6.813.1 Detailed Description

template<typename _Arg, typename _Result>
class std::pointer_to_unary_function< _Arg, _Result >

One of the adaptors for function pointers.

6.813.2 Member Typedef Documentation

argument_type

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary_function< _Arg, _Result >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
template<typename _Arg, typename _Result>
typedef _Result std::unary_function< _Arg, _Result >::result_type  [inherited]
```

`result_type` is the return type
The documentation for this class was generated from the following file:

- stl_function.h

6.814 std::pointer_traits< _Ptr > Struct Template Reference

`#include <memory>`

Public Types

- using difference_type
- using element_type
- using pointer
- template<typename _Up>
  using rebind

Static Public Member Functions

- static constexpr pointer pointer_to (element_type &__r)

6.814.1 Detailed Description

template<typename _Ptr>
struct std::pointer_traits< _Ptr >

Uniform interface to all pointer-like types.

Since

C++11

6.814.2 Member Typedef Documentation

difference_type

```
template<typename _Ptr, typename _Elt>
using std::__ptr_traits_impl< _Ptr, _Elt >::difference_type  [inherited]
```

The type used to represent the difference between two pointers.

element_type

```
template<typename _Ptr, typename _Elt>
using std::__ptr_traits_impl< _Ptr, _Elt >::element_type  [inherited]
```

The type pointed to.

pointer

```
template<typename _Ptr, typename _Elt>
using std::__ptr_traits_impl< _Ptr, _Elt >::pointer  [inherited]
```

The pointer type.

rebind

```
template<typename _Ptr, typename _Elt>
template<typename _Up>
using std::__ptr_traits_impl< _Ptr, _Elt >::rebind  [inherited]
```

A pointer to a different type.

6.814.3 Member Function Documentation

pointer_to()

```
template<typename _Ptr, typename _Elt, bool = is_void<_Elt>::value>
constexpr pointer std::__ptr_traits_ptr_to< _Ptr, _Elt, bool >::pointer_to (
            element_type & __r)  [inline], [static], [constexpr], [inherited]
```

Obtain a pointer to an object.

Parameters

↩ _↩ ↩ _↩ *r*	A reference to an object of type `element_type`

Returns

`pointer::pointer_to(__r)`

Precondition

`pointer::pointer_to(__r)` is a valid expression.

The documentation for this struct was generated from the following file:

- ptr_traits.h

6.815 std::pointer_traits< _Tp * > Struct Template Reference

`#include <memory>`

Public Types

- typedef ptrdiff_t difference_type
- using difference_type
- typedef _Tp element_type
- using element_type
- typedef _Tp * pointer
- using pointer
- template<typename _Up>
  using rebind
- using rebind

Static Public Member Functions

- static constexpr pointer pointer_to (element_type &__r)
- static constexpr pointer pointer_to (element_type &__r)
- static constexpr pointer pointer_to (element_type &__r)

6.815.1 Detailed Description

template<typename _Tp>
struct std::pointer_traits< _Tp * >

Partial specialization for built-in pointers.

Since

C++11

6.815.2 Member Typedef Documentation

difference_type [1/2]

```
template<typename _Tp>
typedef ptrdiff_t std::pointer_traits< _Tp * >::difference_type
```

Type used to represent the difference between two pointers.

difference_type [2/2]

```
using std::__ptr_traits_impl< _Tp *, _Elt >::difference_type
```

The type used to represent the difference between two pointers.

element_type [1/2]

```
template<typename _Tp>
typedef _Tp std::pointer_traits< _Tp * >::element_type
```

The type pointed to.

element_type [2/2]

```
using std::__ptr_traits_impl< _Tp *, _Elt >::element_type
```

The type pointed to.

pointer [1/2]

```
template<typename _Tp>
typedef _Tp* std::pointer_traits< _Tp * >::pointer
```

The pointer type.

pointer [2/2]

```
using std::__ptr_traits_impl< _Tp *, _Elt >::pointer
```

The pointer type.

rebind [1/2]

```
template<typename _Tp>
template<typename _Up>
using std::pointer_traits< _Tp * >::rebind
```

A pointer to a different type.

rebind [2/2]

```
using std::__ptr_traits_impl< _Tp *, _Elt >::rebind
```

A pointer to a different type.

6.815.3 Member Function Documentation

pointer_to() [1/3]

```
template<typename _Ptr, typename _Elt, bool = is_void<_Elt>::value>
constexpr pointer std::__ptr_traits_ptr_to< _Ptr, _Elt, bool >::pointer_to (
            element_type & __r)  [inline], [static], [constexpr], [inherited]
```

Obtain a pointer to an object.

Parameters

↩ _↩ ↩ _↩ *r*	A reference to an object of type `element_type`

Returns

`pointer::pointer_to(__r)`

Precondition

`pointer::pointer_to(__r)` is a valid expression.

pointer_to() [2/3]

```
template<typename _Ptr, typename _Elt, bool = is_void<_Elt>::value>
constexpr pointer std::__ptr_traits_ptr_to< _Ptr, _Elt, bool >::pointer_to (
            element_type & __r)  [inline], [static], [constexpr], [inherited]
```

Obtain a pointer to an object.

Parameters

↩ _↩ ↩ _↩ *r*	A reference to an object of type `element_type`

Returns

`pointer::pointer_to(__r)`

Precondition

`pointer::pointer_to(__r)` is a valid expression.

pointer_to() [3/3]

```
constexpr pointer std::__ptr_traits_ptr_to< _Tp *, _Elt, bool >::pointer_to (
            element_type & __r)  [inline], [static], [constexpr]
```

Obtain a pointer to an object.

Parameters

↩ _↩ ↩ _↩ *r*	A reference to an object of type `element_type`

Returns

`pointer::pointer_to(__r)`

Precondition

`pointer::pointer_to(__r)` is a valid expression.

The documentation for this struct was generated from the following file:

- ptr_traits.h

6.816 std::poisson_distribution< _IntType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- **poisson_distribution** (const param_type &__p)
- **poisson_distribution** (double __mean)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type max () const
- double mean () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::poisson_distribution< _IntType1 > &__x)
- bool operator== (const poisson_distribution &__d1, const poisson_distribution &__d2)
- template<typename _IntType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::poisson_distribution< _IntType1 > &__x)

6.816.1 Detailed Description

template<typename _IntType = int>
class std::poisson_distribution< _IntType >

A discrete Poisson random number distribution.
The formula for the Poisson probability density function is $p(i|\mu) = \frac{\mu^i}{i!}e^{-\mu}$ where μ is the parameter of the distribution.

Since

C++11

6.816.2 Member Typedef Documentation

result_type

```
template<typename _IntType = int>
typedef _IntType std::poisson_distribution< _IntType >::result_type
```

The type of the range of the distribution.

6.816.3 Member Function Documentation

max()

```
template<typename _IntType = int>
result_type std::poisson_distribution< _IntType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

mean()

```
template<typename _IntType = int>
double std::poisson_distribution< _IntType >::mean () const  [inline]
```

Returns the distribution parameter mean.

min()

```
template<typename _IntType = int>
result_type std::poisson_distribution< _IntType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()() [1/2]

```
template<typename _IntType = int>
template<typename _UniformRandomNumberGenerator>
result_type std::poisson_distribution< _IntType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

operator()() [2/2]

```
template<typename _IntType>
template<typename _UniformRandomNumberGenerator>
poisson_distribution< _IntType >::result_type std::poisson_distribution< _IntType >::operator()
(
            _UniformRandomNumberGenerator & __urng,
            const param_type & __param)
```

A rejection algorithm when mean >= 12 and a simple method based upon the multiplication of uniform random variates otherwise. NB: The former is available only if _GLIBCXX_USE_C99_MATH_FUNCS is defined.
Reference: Devroye, L. Non-Uniform Random Variates Generation. Springer-Verlag, New York, 1986, Ch. X, Sects. 3.3 & 3.4 (+ Errata!).
References std::abs(), std::numeric_limits< _Tp >::epsilon(), std::log(), and std::numeric_limits< _Tp >::max().

param() [1/2]

```
template<typename _IntType = int>
param_type std::poisson_distribution< _IntType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _IntType = int>
void std::poisson_distribution< _IntType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _IntType = int>
void std::poisson_distribution< _IntType >::reset ()  [inline]
```

Resets the distribution state.

6.816.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::poisson_distribution< _IntType1 > & __x)  [friend]
```

Inserts a poisson_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A poisson_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _IntType = int>
bool operator== (
            const poisson_distribution< _IntType > & __d1,
            const poisson_distribution< _IntType > & __d2)  [friend]
```

Return true if two Poisson distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _IntType = int>
template<typename _IntType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::poisson_distribution< _IntType1 > & __x)  [friend]
```

Extracts a poisson_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _*is*	An input stream.
_↩ _*x*	A poisson_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.817 std::pmr::polymorphic_allocator< _Tp > Class Template Reference

`#include <memory_resource>`

Public Types

- using **value_type**

Public Member Functions

- **polymorphic_allocator** (const polymorphic_allocator &__other)=default
- template<typename _Up>
  polymorphic_allocator (const polymorphic_allocator< _Up > &__x) noexcept
- **polymorphic_allocator** (memory_resource *__r) noexcept
- _Tp * **allocate** (size_t __n)
- template<typename _Tp1, typename... _Args>
  void **construct** (_Tp1 *__p, _Args &&... __args)
- void **deallocate** (_Tp *__p, size_t __n) noexcept
- template<typename _Up>
  void **destroy** (_Up *__p)
- polymorphic_allocator & **operator=** (const polymorphic_allocator &)=delete
- memory_resource * **resource** () const noexcept
- polymorphic_allocator **select_on_container_copy_construction** () const noexcept

Friends

- bool **operator==** (const polymorphic_allocator &__a, const polymorphic_allocator &__b) noexcept

6.817.1 Detailed Description

template<typename _Tp>
class std::pmr::polymorphic_allocator< _Tp >

Class template polymorphic_allocator.

Since

C++17

The documentation for this class was generated from the following file:

- memory_resource.h

6.818 std::pmr::pool_options Struct Reference

`#include <memory_resource>`

Public Attributes

- size_t **largest_required_pool_block**
- size_t max_blocks_per_chunk

6.818.1 Detailed Description

Parameters for tuning a pool resource's behaviour.

Since

C++17

6.818.2 Member Data Documentation

max_blocks_per_chunk

`size_t std::pmr::pool_options::max_blocks_per_chunk`
Upper limit on number of blocks in a chunk.
A lower value prevents allocating huge chunks that could remain mostly unused, but means pools will need to replenished more frequently.
The documentation for this struct was generated from the following file:

- memory_resource

6.819 __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc > Class Template Reference

`#include <priority_queue.hpp>`

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef __rebind_va::const_pointer **const_pointer**
- typedef __rebind_va::const_reference **const_reference**
- typedef Tag **container_category**
- typedef allocator_type::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef __rebind_va::pointer **pointer**
- typedef __rebind_va::reference **reference**
- typedef allocator_type::size_type **size_type**
- typedef _Tv **value_type**

Public Member Functions

- priority_queue (const cmp_fn &r_cmp_fn)
- **priority_queue** (const priority_queue &other)
- template<typename It>
  priority_queue (It first_it, It last_it)
- template<typename It>
  priority_queue (It first_it, It last_it, const cmp_fn &r_cmp_fn)
- priority_queue & **operator=** (const priority_queue &other)
- void **swap** (priority_queue &other)

6.819.1 Detailed Description

template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename _Alloc = std::allocator<char>>
class __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >

A priority queue composed of one specific heap policy.

Template Parameters

_Tv	Value type.
Cmp_Fn	Comparison functor.
Tag	Instantiating data structure type, see container_tag.
_Alloc	Allocator type.

Base is dispatched at compile time via Tag, from the following choices: binary_heap_tag, binomial_heap_tag, pairing↩
_heap_tag, rc_binomial_heap_tag, thin_heap_tag
Base choices are: detail::binary_heap, detail::binomial_heap, detail::pairing_heap, detail::rc_binomial_heap, detail::↩
thin_heap.

6.819.2 Constructor & Destructor Documentation

priority_queue() [1/3]

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
            const cmp_fn & r_cmp_fn)  [inline]
```

Constructor taking some policy objects. r_cmp_fn will be copied by the Cmp_Fn object of the container object.

priority_queue() [2/3]

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
            It first_it,
            It last_it)  [inline]
```

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

priority_queue() [3/3]

```
template<typename _Tv, typename Cmp_Fn = std::less<_Tv>, typename Tag = pairing_heap_tag, typename
_Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >::priority_queue (
            It first_it,
            It last_it,
            const cmp_fn & r_cmp_fn)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_cmp_fn will be copied by the cmp_fn object of the container object.
The documentation for this class was generated from the following file:

- priority_queue.hpp

6.820 std::priority_queue< _Tp, _Sequence, _Compare > Class Template Reference

`#include <stl_queue.h>`

Public Types

- typedef _Sequence::const_reference **const_reference**
- typedef _Sequence **container_type**
- typedef _Sequence::reference **reference**
- typedef _Sequence::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Sequence::value_type **value_type**

Public Member Functions

- template<typename _Seq = _Sequence, typename _Requires = typename enable_if<__and_<is_default_constructible<_Compare>, is↩ _default_constructible<_Seq>>::value>::type>
  constexpr priority_queue ()
- template<typename _InputIterator, typename _Alloc, typename = std::_RequireInputIter<_InputIterator>, typename _Requires = _Uses<↩ _Alloc>>
  constexpr **priority_queue** (_InputIterator __first, _InputIterator __last, const _Alloc &__alloc)
- template<typename _InputIterator, typename _Alloc, typename _Requires = _Uses<_Alloc>>
  constexpr **priority_queue** (_InputIterator __first, _InputIterator __last, const _Compare &__x, _Sequence &&_↩ _s, const _Alloc &__alloc)
- template<typename _InputIterator, typename _Alloc, typename = std::_RequireInputIter<_InputIterator>, typename _Requires = _Uses<↩ _Alloc>>
  constexpr **priority_queue** (_InputIterator __first, _InputIterator __last, const _Compare &__x, const _Alloc &_↩ _alloc)
- template<typename _InputIterator, typename _Alloc, typename = std::_RequireInputIter<_InputIterator>, typename _Requires = _Uses<↩ _Alloc>>
  constexpr **priority_queue** (_InputIterator __first, _InputIterator __last, const _Compare &__x, const _Sequence &__s, const _Alloc &__alloc)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  constexpr **priority_queue** (const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  constexpr **priority_queue** (const _Compare &__x, _Sequence &&__c, const _Alloc &__a)
- constexpr **priority_queue** (const _Compare &__x, _Sequence &&__s=_Sequence())
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  constexpr **priority_queue** (const _Compare &__x, const _Alloc &__a)

- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  constexpr **priority_queue** (const _Compare &__x, const _Sequence &__c, const _Alloc &__a)
- constexpr **priority_queue** (const _Compare &__x, const _Sequence &__s)
- **priority_queue** (const priority_queue &)=default
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  constexpr **priority_queue** (const priority_queue &__q, const _Alloc &__a)
- constexpr **priority_queue** (priority_queue &&__q) noexcept(__and_< is_nothrow_move_constructible< _Sequence >, is_nothrow_move_constructible< _Compare > >::value)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  constexpr **priority_queue** (priority_queue &&__q, const _Alloc &__a)
- template<typename... _Args>
  constexpr void **emplace** (_Args &&... __args)
- constexpr bool empty () const
- priority_queue & **operator=** (const priority_queue &)=default
- constexpr priority_queue & **operator=** (priority_queue &&__q) noexcept(__and_< is_nothrow_move_assignable< _Sequence >, is_nothrow_move_assignable< _Compare > >::value)
- constexpr void pop ()
- constexpr void push (const value_type &__x)
- constexpr void **push** (value_type &&__x)
- constexpr size_type size () const
- constexpr void **swap** (priority_queue &__pq) noexcept(__and_<//c++1z or gnu++11 __is_nothrow_swappable< _Sequence >, __is_nothrow_swappable< _Compare > >::value)
- constexpr const_reference top () const

- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x=_Compare())
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x, const _Sequence &__s)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr priority_queue (_InputIterator __first, _InputIterator __last, const _Compare &__x, _Sequence &&__s)

Protected Attributes

- _Sequence **c**
- _Compare **comp**

6.820.1 Detailed Description

template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _Sequence::value_type>>
class std::priority_queue< _Tp, _Sequence, _Compare >

A standard container automatically sorting its contents.

Template Parameters

_Tp	Type of element.
_Sequence	Type of underlying sequence, defaults to vector<_Tp>.
_Compare	Comparison function object type, defaults to less<_Sequence::value_type>.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces priority-based sorting and queue behavior. Very few of the standard container/sequence interface requirements are met (e.g., iterators).
The second template parameter defines the type of the underlying sequence/container. It defaults to std::vector, but it can be any type that supports `front()`, `push_back`, `pop_back`, and random-access iterators, such as std::deque or an appropriate user-defined type.
The third template parameter supplies the means of making priority comparisons. It defaults to `less<value_type>` but can be anything defining a strict weak ordering.
Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard queue operations.

Note

No equality/comparison operators are provided for priority_queue.

Sorting of the elements takes place as they are added to, and removed from, the priority_queue using the priority_queue's member functions. If you access the elements by other means, and change their data such that the sorting order would be different, the priority_queue will not re-sort the elements for you. (How could it know to do so?)

6.820.2 Constructor & Destructor Documentation

priority_queue() [1/4]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<__and_<is_default_↩
constructible<_Compare>, is_default_constructible<_Seq>>::value>::type>
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue ()  [inline], [constexpr]
```

Default constructor creates no elements.

priority_queue() [2/4]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue (
            _InputIterator __first,
            _InputIterator __last,
            const _Compare & __x = _Compare())  [inline], [constexpr]
```

Builds a queue from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__x	A comparison functor describing a strict weak ordering.
__s	An initial sequence with which to start.

Begins by copying *__s*, inserting a copy of the elements from [first,last) into the copy of *__s*, then ordering the copy according to *__x*.
For more information on function objects, see the documentation on functor base classes.

priority_queue() [3/4]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue (
            _InputIterator __first,
            _InputIterator __last,
            const _Compare & __x,
            const _Sequence & __s)  [inline], [constexpr]
```

Builds a queue from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__x	A comparison functor describing a strict weak ordering.
__s	An initial sequence with which to start.

Begins by copying *__s*, inserting a copy of the elements from [first,last) into the copy of *__s*, then ordering the copy according to *__x*.
For more information on function objects, see the documentation on functor base classes.

priority_queue() [4/4]

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::priority_queue< _Tp, _Sequence, _Compare >::priority_queue (
            _InputIterator __first,
            _InputIterator __last,
            const _Compare & __x,
            _Sequence && __s)  [inline], [constexpr]
```

Builds a queue from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__x	A comparison functor describing a strict weak ordering.
__s	An initial sequence with which to start.

Begins by copying *__s*, inserting a copy of the elements from [first,last) into the copy of *__s*, then ordering the copy according to *__x*.
For more information on function objects, see the documentation on functor base classes.
References std::move().

6.820.3 Member Function Documentation

empty()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
bool std::priority_queue< _Tp, _Sequence, _Compare >::empty () const  [inline], [nodiscard],
[constexpr]
```

Returns true if the queue is empty.
Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().

pop()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
void std::priority_queue< _Tp, _Sequence, _Compare >::pop ()  [inline], [constexpr]
```

Removes first element.
This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.
Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.
Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().

push()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
void std::priority_queue< _Tp, _Sequence, _Compare >::push (
            const value_type & __x)  [inline], [constexpr]
```

Add data to the queue.

Parameters

_↩ _x	Data to be added.

This is a typical queue operation. The time complexity of the operation depends on the underlying sequence.
Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().

size()

```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
size_type std::priority_queue< _Tp, _Sequence, _Compare >::size () const  [inline], [nodiscard],
[constexpr]
```

Returns the number of elements in the queue.

top()

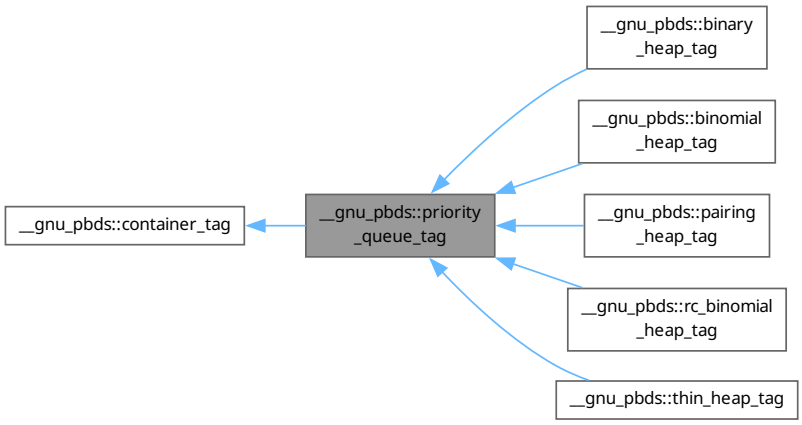
```
template<typename _Tp, typename _Sequence = vector<_Tp>, typename _Compare = less<typename _↩
Sequence::value_type>>
const_reference std::priority_queue< _Tp, _Sequence, _Compare >::top () const  [inline], [nodiscard],
[constexpr]
```

Returns a read-only (constant) reference to the data at the first element of the queue.
Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().
The documentation for this class was generated from the following file:

- stl_queue.h

6.821 __gnu_pbds::priority_queue_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::priority_queue_tag:



6.821.1 Detailed Description

Basic priority-queue.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.822 __gnu_pbds::detail::probe_fn_base< _Alloc > Class Template Reference

`#include <probe_fn_base.hpp>`

6.822.1 Detailed Description

template<typename _Alloc>
class __gnu_pbds::detail::probe_fn_base< _Alloc >

Probe functor base.
The documentation for this class was generated from the following file:

- probe_fn_base.hpp

6.823 __gnu_cxx::project1st< _Arg1, _Arg2 > Struct Template Reference

`#include <functional>`

Public Types

- typedef _Arg1 first_argument_type
- typedef _Arg1 result_type
- typedef _Arg2 second_argument_type

Public Member Functions

- _Arg1 **operator()** (const _Arg1 &__x, const _Arg2 &) const

6.823.1 Detailed Description

template<class _Arg1, class _Arg2>
struct __gnu_cxx::project1st< _Arg1, _Arg2 >

An SGI extension .

6.823.2 Member Typedef Documentation

first_argument_type

```
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Arg1 >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

```
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Arg1 >::result_type  [inherited]
```

`result_type` is the return type

second_argument_type

```
typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Arg1 >::second_argument_type  [inherited]
```

`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- ext/functional

6.824 __gnu_cxx::project2nd< _Arg1, _Arg2 > Struct Template Reference

```
#include <functional>
```

Public Types

- typedef _Arg1 first_argument_type
- typedef _Arg2 result_type
- typedef _Arg2 second_argument_type

Public Member Functions

- _Arg2 **operator()** (const _Arg1 &, const _Arg2 &__y) const

6.824.1 Detailed Description

template<class _Arg1, class _Arg2>
struct __gnu_cxx::project2nd< _Arg1, _Arg2 >

An SGI extension .

6.824.2 Member Typedef Documentation

first_argument_type

```
typedef _Arg1 std::binary_function< _Arg1, _Arg2, _Arg2 >::first_argument_type  [inherited]
```

`first_argument_type` is the type of the first argument

result_type

`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Arg2 >::result_type [inherited]`
`result_type` is the return type

second_argument_type

`typedef _Arg2 std::binary_function< _Arg1, _Arg2, _Arg2 >::second_argument_type [inherited]`
`second_argument_type` is the type of the second argument
The documentation for this struct was generated from the following file:

- ext/functional

6.825 std::promise< _Res > Class Template Reference

`#include <future>`

Public Member Functions

- template<typename _Allocator>
  promise (allocator_arg_t, const _Allocator &__a)
- **promise** (const promise &)=delete
- **promise** (promise &&__rhs) noexcept
- future< _Res > **get_future** ()
- promise & **operator=** (const promise &)=delete
- promise & **operator=** (promise &&__rhs) noexcept
- void **set_exception** (exception_ptr __p)
- void **set_exception_at_thread_exit** (exception_ptr __p)
- void **set_value** (_Res &&__r)
- void **set_value** (const _Res &__r)
- void **set_value_at_thread_exit** (_Res &&__r)
- void **set_value_at_thread_exit** (const _Res &__r)
- void **swap** (promise &__rhs) noexcept

Friends

- template<typename, typename>
  struct **_State::_Setter**

6.825.1 Detailed Description

template<typename _Res>
class std::promise< _Res >

Primary template for promise.
The documentation for this class was generated from the following file:

- future

6.826 std::promise< _Res & > Class Template Reference

`#include <future>`

Public Member Functions

- template<typename _Allocator>
  promise (allocator_arg_t, const _Allocator &__a)
- **promise** (const promise &)=delete
- **promise** (promise &&__rhs) noexcept
- future< _Res & > **get_future** ()
- promise & **operator=** (const promise &)=delete
- promise & **operator=** (promise &&__rhs) noexcept
- void **set_exception** (exception_ptr __p)
- void **set_exception_at_thread_exit** (exception_ptr __p)
- void **set_value** (_Res &__r)
- void **set_value_at_thread_exit** (_Res &__r)
- void **swap** (promise &__rhs) noexcept

Friends

- template<typename, typename>
  struct **_State::_Setter**

6.826.1 Detailed Description

template<typename _Res>
class std::promise< _Res & >

Partial specialization for promise<R&>.
The documentation for this class was generated from the following file:

- future

6.827 std::promise< void > Class Reference

`#include <future>`

Public Member Functions

- template<typename _Allocator>
  promise (allocator_arg_t, const _Allocator &__a)
- **promise** (const promise &)=delete
- **promise** (promise &&__rhs) noexcept
- future< void > **get_future** ()
- promise & **operator=** (const promise &)=delete
- promise & **operator=** (promise &&__rhs) noexcept
- void **set_exception** (exception_ptr __p)
- void **set_exception_at_thread_exit** (exception_ptr __p)
- void **set_value** ()
- void **set_value_at_thread_exit** ()
- void **swap** (promise &__rhs) noexcept

Friends

- template<typename, typename>
  struct **_State::_Setter**

6.827.1 Detailed Description

Explicit specialization for promise<void>.
The documentation for this class was generated from the following file:

- future

6.828 std::experimental::fundamentals_v2::propagate_const< _Tp > Class Template Reference

`#include <propagate_const>`

Public Types

- using **element_type**

Public Member Functions

- template<typename _Up, typename enable_if< __and_< is_constructible< _Tp, _Up && >, __not_< is_convertible< _Up &&, _Tp > >, __not_< __is_propagate_const< typename decay< _Up >::type > > >::value, bool >::type = false>
  constexpr **propagate_const** (_Up &&__u)
- template<typename _Up, typename enable_if< __and_< is_constructible< _Tp, _Up && >, is_convertible< _Up &&, _Tp >, __not_< __is_propagate_const< typename decay< _Up >::type > > >::value, bool >::type = true>
  constexpr **propagate_const** (_Up &&__u)
- **propagate_const** (const propagate_const &__p)=delete
- constexpr **propagate_const** (propagate_const &&__p)=default
- template<typename _Up, typename enable_if< __and_< is_constructible< _Tp, _Up && >, __not_< is_convertible< _Up &&, _Tp > > >::value, bool >::type = false>
  constexpr **propagate_const** (propagate_const< _Up > &&__pu)
- template<typename _Up, typename enable_if< __and_< is_constructible< _Tp, _Up && >, is_convertible< _Up &&, _Tp > >::value, bool >::type = true>
  constexpr **propagate_const** (propagate_const< _Up > &&__pu)
- constexpr element_type * **get** ()
- constexpr const element_type * **get** () const
- constexpr **operator bool** () const
- constexpr element_type & **operator*** ()
- constexpr const element_type & **operator*** () const
- constexpr element_type * **operator->** ()
- constexpr const element_type * **operator->** () const
- template<typename _Up, typename = typename enable_if<__and_<is_convertible<_Up&&, _Tp>, __not_<__is_propagate_const< typename decay<_Up>::type>> >::value>::type>
  constexpr propagate_const & **operator=** (_Up &&__u)
- propagate_const & **operator=** (const propagate_const &__p)=delete
- constexpr propagate_const & **operator=** (propagate_const &&__p)=default
- template<typename _Up, typename = typename enable_if<is_convertible<_Up&&, _Tp>::value>::type>
  constexpr propagate_const & **operator=** (propagate_const< _Up > &&__pu)
- constexpr void **swap** (propagate_const &__pt) noexcept(__is_nothrow_swappable< _Tp >::value)

Friends

- template<typename _Up>
  constexpr const _Up & **get_underlying** (const propagate_const< _Up > &__pt) noexcept
- template<typename _Up>
  constexpr _Up & **get_underlying** (propagate_const< _Up > &__pt) noexcept

6.828.1 Detailed Description

template<typename _Tp>
class std::experimental::fundamentals_v2::propagate_const< _Tp >

Const-propagating wrapper.
The documentation for this class was generated from the following file:

- propagate_const

6.829 __gnu_pbds::quadratic_probe_fn< Size_Type > Class Template Reference

`#include <hash_policy.hpp>`

Public Types

- typedef Size_Type **size_type**

Public Member Functions

- void **swap** (quadratic_probe_fn< Size_Type > &other)

Protected Member Functions

- size_type operator() (size_type i) const

6.829.1 Detailed Description

template<typename Size_Type = std::size_t>
class __gnu_pbds::quadratic_probe_fn< Size_Type >

A probe sequence policy using square increments.

6.829.2 Member Function Documentation

operator()()

```
template<typename Size_Type>
quadratic_probe_fn< Size_Type >::size_type __gnu_pbds::quadratic_probe_fn< Size_Type >::operator()
(
            size_type i) const  [inline], [protected]
```

Returns the i-th offset from the hash value.
The documentation for this class was generated from the following file:

- hash_policy.hpp

6.830 std::queue< _Tp, _Sequence > Class Template Reference

`#include <stl_queue.h>`

Public Types

- typedef _Sequence::const_reference **const_reference**
- typedef _Sequence **container_type**
- typedef _Sequence::reference **reference**
- typedef _Sequence::size_type **size_type**
- typedef _Sequence::value_type **value_type**

Public Member Functions

- template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<_Seq>::value>::type>
  queue ()
- **queue** (_Sequence &&__c)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  queue (_Sequence &&__c, const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  queue (const _Alloc &__a)
- **queue** (const _Sequence &__c)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  queue (const _Sequence &__c, const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  queue (const queue &__q, const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  queue (queue &&__q, const _Alloc &__a)
- reference back ()
- const_reference back () const
- template<typename... _Args>
  decltype(auto) **emplace** (_Args &&... __args)
- bool empty () const
- reference front ()
- const_reference front () const
- void pop ()
- void push (const value_type &__x)
- void **push** (value_type &&__x)
- size_type size () const
- void **swap** (queue &__q) noexcept(__is_nothrow_swappable< _Sequence >::value)

Protected Attributes

- _Sequence c

Friends

- template<typename _Tp1, typename _Seq1>
  bool **operator**< (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)
- template<typename _Tp1, three_way_comparable _Seq1>
  compare_three_way_result_t< _Seq1 > **operator**<=> (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)
- template<typename _Tp1, typename _Seq1>
  bool **operator==** (const queue< _Tp1, _Seq1 > &, const queue< _Tp1, _Seq1 > &)

6.830.1 Detailed Description

template<typename _Tp, typename _Sequence = deque<_Tp>>
class std::queue< _Tp, _Sequence >

A standard container giving FIFO behavior.

Template Parameters

_Tp	Type of element.

_Sequence	Type of underlying sequence, defaults to deque<_Tp>.

Meets many of the requirements of a container, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.
This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-first-out queue behavior.
The second template parameter defines the type of the underlying sequence/container. It defaults to std::deque, but it can be any type that supports `front`, `back`, `push_back`, and `pop_front`, such as std::list or an appropriate user-defined type.
Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push` and `pop`, which are standard queue/FIFO operations.

6.830.2 Constructor & Destructor Documentation

queue()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<↩
_Seq>::value>::type>
std::queue< _Tp, _Sequence >::queue ()  [inline]
```

Default constructor creates no elements.
References c.

6.830.3 Member Function Documentation

back() [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue< _Tp, _Sequence >::back ()  [inline], [nodiscard]
```

Returns a read/write reference to the data at the last element of the queue.
References c.

back() [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue< _Tp, _Sequence >::back () const  [inline], [nodiscard]
```

Returns a read-only (constant) reference to the data at the last element of the queue.
References c.

empty()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::queue< _Tp, _Sequence >::empty () const  [inline], [nodiscard]
```

Returns true if the queue is empty.
References c.

front() [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::queue< _Tp, _Sequence >::front ()  [inline], [nodiscard]
```

Returns a read/write reference to the data at the first element of the queue.
References c.

front() [2/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::queue< _Tp, _Sequence >::front () const  [inline], [nodiscard]
```

Returns a read-only (constant) reference to the data at the first element of the queue.
References c.

pop()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue< _Tp, _Sequence >::pop ()  [inline]
```

Removes first element.
This is a typical queue operation. It shrinks the queue by one. The time complexity of the operation depends on the underlying sequence.
Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.
References c.

push()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::queue< _Tp, _Sequence >::push (
            const value_type & __x)  [inline]
```

Add data to the end of the queue.

Parameters

_↩ _x	Data to be added.

This is a typical queue operation. The function creates an element at the end of the queue and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.
References c.

size()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::queue< _Tp, _Sequence >::size () const  [inline], [nodiscard]
```

Returns the number of elements in the queue.
References c.

6.830.4 Member Data Documentation

c

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
_Sequence std::queue< _Tp, _Sequence >::c  [protected]
```

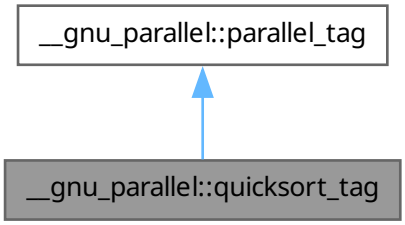
c is the underlying container.
Referenced by queue(), back(), back(), empty(), front(), front(), std::operator<(), std::operator==(), pop(), push(), and size().
The documentation for this class was generated from the following file:

- stl_queue.h

6.831 __gnu_parallel::quicksort_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::quicksort_tag:



Public Member Functions

- **quicksort_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

6.831.1 Detailed Description

Forces parallel sorting using unbalanced quicksort at compile time.

6.831.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

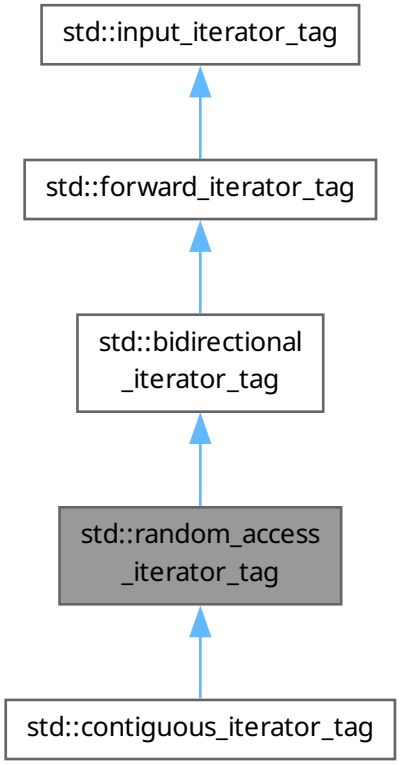
__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

6.832 std::random_access_iterator_tag Struct Reference

`#include <stl_iterator_base_types.h>`
Inheritance diagram for std::random_access_iterator_tag:



6.832.1 Detailed Description

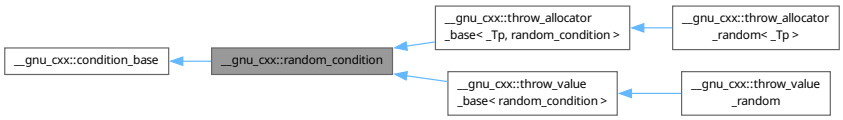
Random-access iterators support a superset of bidirectional iterator operations.
The documentation for this struct was generated from the following file:

- stl_iterator_base_types.h

6.833 __gnu_cxx::random_condition Struct Reference

`#include <throw_allocator.h>`

Inheritance diagram for __gnu_cxx::random_condition:



Classes

- struct always_adjustor
- struct group_adjustor
- struct never_adjustor

Public Member Functions

- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

6.833.1 Detailed Description

Base class for random probability control and throw.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.834 std::random_device Class Reference

`#include <random>`

Public Types

- typedef unsigned int result_type

Public Member Functions

- **random_device** (const random_device &)=delete
- **random_device** (const std::string &__token)
- double **entropy** () const noexcept
- result_type **operator()** ()
- void **operator=** (const random_device &)=delete

Static Public Member Functions

- static constexpr result_type **max** ()
- static constexpr result_type **min** ()

6.834.1 Detailed Description

A standard interface to a platform-specific non-deterministic random number generator (if any are available).

Since

C++11

6.834.2 Member Typedef Documentation

result_type

```
typedef unsigned int std::random_device::result_type
```

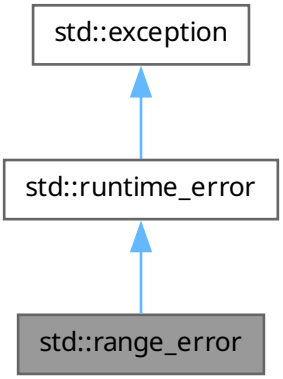
The type of the generated random value.
The documentation for this class was generated from the following file:

- random.h

6.835 std::range_error Class Reference

```
#include <stdexcept>
```

Inheritance diagram for std::range_error:



Public Member Functions

- **range_error** (const char *)
- **range_error** (const range_error &)=default
- **range_error** (const string &__arg)
- **range_error** (range_error &&)=default
- range_error & **operator=** (const range_error &)=default
- range_error & **operator=** (range_error &&)=default
- virtual const char * what () const noexcept

6.835.1 Detailed Description

Thrown to indicate range errors in internal computations.

6.835.2 Member Function Documentation

what()

```
virtual const char * std::runtime_error::what () const  [virtual], [noexcept], [inherited]
```

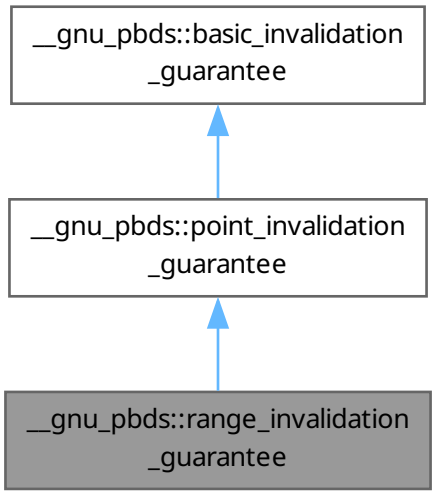
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::experimental::filesystem::v1::filesystem_error, std::filesystem::filesystem_error, and std::ios_base::failure.
References what().
Referenced by what().
The documentation for this class was generated from the following file:

- stdexcept

6.836 __gnu_pbds::range_invalidation_guarantee Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::range_invalidation_guarantee:



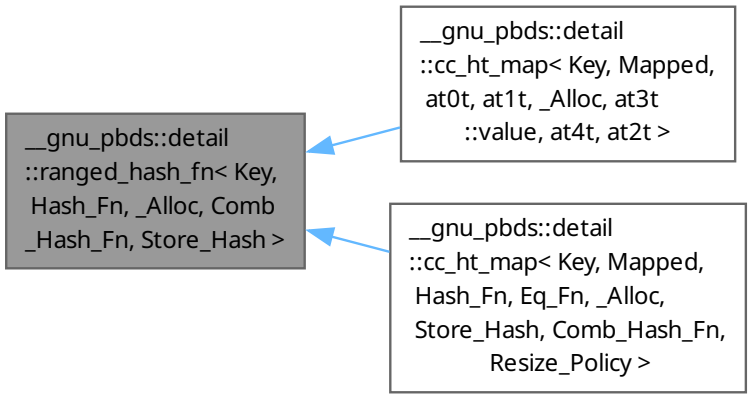
6.836.1 Detailed Description

Signifies an invalidation guarantee that includes all those of its base, and additionally, that any range-type iterator (including the returns of begin() and end()) is in the correct relative positions to other range-type iterators as long as its corresponding entry has not be erased, regardless of modifications to the container object.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.837 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >:



6.837.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn, bool Store_Hash>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >

Primary template.
The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

6.838 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false > Class Template Reference

`#include <ranged_hash_fn.hpp>`

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef Hash_Fn **hash_fn_base**
- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Hash_Fn &)
- **ranged_hash_fn** (size_type, const Hash_Fn &, const Comb_Hash_Fn &)
- void **notify_resized** (size_type)
- size_type **operator()** (key_const_reference) const
- void **swap** (ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false > &)

6.838.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >

Specialization 1 The client supplies a hash function and a ranged hash function, and requests that hash values not be stored.
The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

6.839 __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true > Class Template Reference

`#include <ranged_hash_fn.hpp>`

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef Hash_Fn **hash_fn_base**
- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Hash_Fn &)
- **ranged_hash_fn** (size_type, const Hash_Fn &, const Comb_Hash_Fn &)
- void **notify_resized** (size_type)
- comp_hash **operator()** (key_const_reference) const
- comp_hash **operator()** (key_const_reference, size_type) const
- void **swap** (ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true > &)

6.839.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >

Specialization 2 The client supplies a hash function and a ranged hash function, and requests that hash values be stored.
The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

6.840 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false > Class Template Reference

`#include <ranged_hash_fn.hpp>`

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Comb_Hash_Fn &)
- **ranged_hash_fn** (size_type, const null_type &, const Comb_Hash_Fn &)
- void **swap** (ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false > &)

6.840.1 Detailed Description

template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >

Specialization 3 The client does not supply a hash function (by specifying null_type as the Hash_Fn parameter), and requests that hash values not be stored.
The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

6.841 __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true > Class Template Reference

`#include <ranged_hash_fn.hpp>`

Protected Types

- typedef Comb_Hash_Fn **comb_hash_fn_base**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_hash_fn** (size_type)
- **ranged_hash_fn** (size_type, const Comb_Hash_Fn &)
- **ranged_hash_fn** (size_type, const null_type &, const Comb_Hash_Fn &)
- void **swap** (ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true > &)

6.841.1 Detailed Description

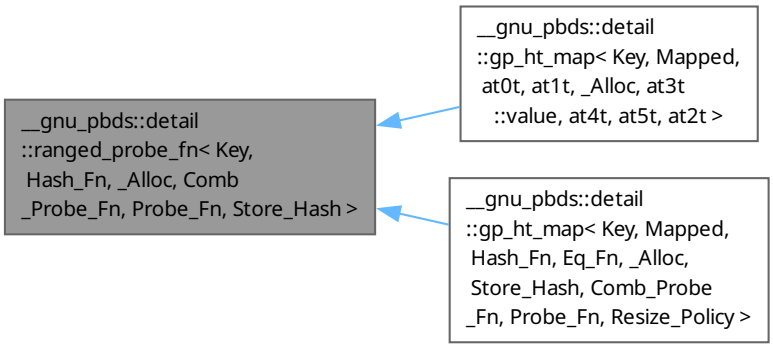
template<typename Key, typename _Alloc, typename Comb_Hash_Fn>
class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >

Specialization 4 The client does not supply a hash function (by specifying null_type as the Hash_Fn parameter), and requests that hash values be stored.
The documentation for this class was generated from the following file:

- ranged_hash_fn.hpp

6.842 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash > Class Template Reference

Inheritance diagram for __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >:



6.842.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn, bool Store_Hash>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >

Primary template.
The documentation for this class was generated from the following file:

- ranged_probe_fn.hpp

6.843 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false > Class Template Reference

`#include <ranged_probe_fn.hpp>`

Protected Types

- typedef Comb_Probe_Fn **comb_probe_fn_base**
- typedef Hash_Fn **hash_fn_base**
- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**
- typedef Probe_Fn **probe_fn_base**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_probe_fn** (size_type)
- **ranged_probe_fn** (size_type, const Hash_Fn &)
- **ranged_probe_fn** (size_type, const Hash_Fn &, const Comb_Probe_Fn &)
- **ranged_probe_fn** (size_type, const Hash_Fn &, const Comb_Probe_Fn &, const Probe_Fn &)

- void **notify_resized** (size_type)
- size_type **operator()** (key_const_reference) const
- size_type **operator()** (key_const_reference, size_type, size_type) const
- void **swap** (ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false > &)

6.843.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >

Specialization 1
The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.
The documentation for this class was generated from the following file:

- ranged_probe_fn.hpp

6.844 __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > Class Template Reference

`#include <ranged_probe_fn.hpp>`

Protected Types

- typedef Comb_Probe_Fn **comb_probe_fn_base**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef Hash_Fn **hash_fn_base**
- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**
- typedef Probe_Fn **probe_fn_base**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_probe_fn** (size_type)
- **ranged_probe_fn** (size_type, const Hash_Fn &)
- **ranged_probe_fn** (size_type, const Hash_Fn &, const Comb_Probe_Fn &)
- **ranged_probe_fn** (size_type, const Hash_Fn &, const Comb_Probe_Fn &, const Probe_Fn &)
- void **notify_resized** (size_type)
- comp_hash **operator()** (key_const_reference) const
- size_type **operator()** (key_const_reference, size_type) const
- size_type **operator()** (key_const_reference, size_type, size_type) const
- void **swap** (ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true > &)

6.844.1 Detailed Description

template<typename Key, typename Hash_Fn, typename _Alloc, typename Comb_Probe_Fn, typename Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >

Specialization 2- The client supplies a probe function and a ranged probe function, and requests that hash values not be stored.
The documentation for this class was generated from the following file:

- ranged_probe_fn.hpp

6.845 __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false > Class Template Reference

`#include <ranged_probe_fn.hpp>`

Protected Types

- typedef Comb_Probe_Fn **comb_probe_fn_base**
- typedef rebind_traits< _Alloc, Key >::const_reference **key_const_reference**
- typedef _Alloc::size_type **size_type**

Protected Member Functions

- **ranged_probe_fn** (size_type size)
- **ranged_probe_fn** (size_type, const Comb_Probe_Fn &r_comb_probe_fn)
- **ranged_probe_fn** (size_type, const null_type &, const Comb_Probe_Fn &r_comb_probe_fn, const null_type &)
- void **swap** (ranged_probe_fn &other)

6.845.1 Detailed Description

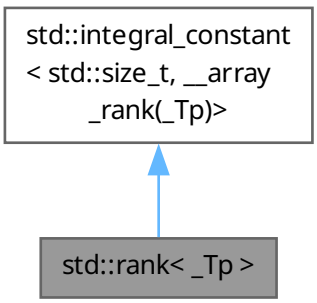
template<typename Key, typename _Alloc, typename Comb_Probe_Fn>
class __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >

Specialization 3 and 4 The client does not supply a hash function or probe function, and requests that hash values not be stored.
The documentation for this class was generated from the following file:

- ranged_probe_fn.hpp

6.846 std::rank< _Tp > Struct Template Reference

`#include <type_traits>`
Inheritance diagram for std::rank< _Tp >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr std::size_t **value**

6.846.1 Detailed Description

template<**typename _Tp**>
struct std::rank< **_Tp** >

rank
The documentation for this struct was generated from the following file:

- type_traits

6.847 std::ratio< _Num, _Den > Struct Template Reference

`#include <ratio>`

Public Types

- typedef ratio< num, den > **type**

Static Public Attributes

- static constexpr intmax_t **den**
- static constexpr intmax_t **num**

6.847.1 Detailed Description

template<**intmax_t _Num, intmax_t _Den = 1**>
struct std::ratio< **_Num, _Den** >

Provides compile-time rational arithmetic.
This class template represents any finite rational number with a numerator and denominator representable by compile-time constants of type intmax_t. The ratio is simplified when instantiated.
For example:

```
std::ratio<7,-21>::num == -1;
std::ratio<7,-21>::den == 3;
```

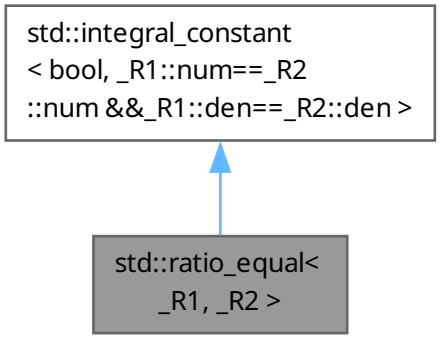
The documentation for this struct was generated from the following file:

- ratio

6.848 std::ratio_equal< _R1, _R2 > Struct Template Reference

`#include <ratio>`

Inheritance diagram for std::ratio_equal< _R1, _R2 >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr bool **value**

6.848.1 Detailed Description

template<typename _R1, typename _R2>
struct std::ratio_equal< _R1, _R2 >

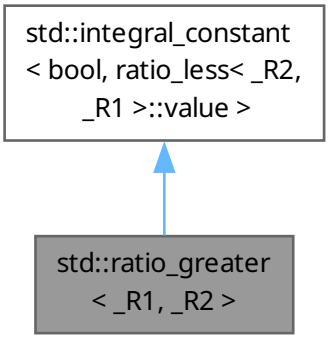
ratio_equal
The documentation for this struct was generated from the following file:

- ratio

6.849 std::ratio_greater< _R1, _R2 > Struct Template Reference

`#include <ratio>`

Inheritance diagram for std::ratio_greater< _R1, _R2 >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr bool **value**

6.849.1 Detailed Description

template<**typename _R1, typename _R2**>
struct std::ratio_greater< **_R1, _R2** >

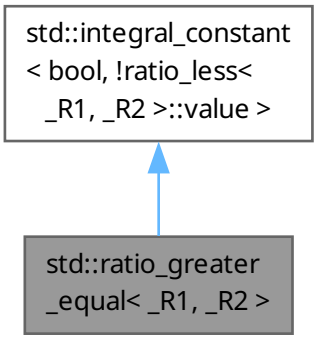
ratio_greater
The documentation for this struct was generated from the following file:

- ratio

6.850 std::ratio_greater_equal< _R1, _R2 > Struct Template Reference

`#include <ratio>`

Inheritance diagram for std::ratio_greater_equal< _R1, _R2 >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr bool **value**

6.850.1 Detailed Description

template<**typename _R1, typename _R2**>
struct std::ratio_greater_equal< **_R1, _R2** >

ratio_greater_equal
The documentation for this struct was generated from the following file:

- ratio

6.851 std::ratio_less< _R1, _R2 > Struct Template Reference

`#include <ratio>`

6.851.1 Detailed Description

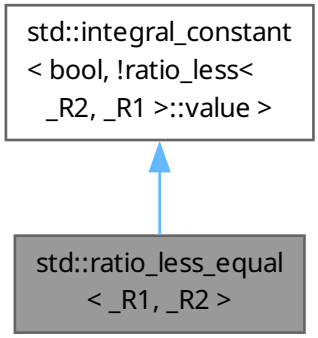
template<**typename _R1, typename _R2**>
struct std::ratio_less< **_R1, _R2** >

ratio_less
The documentation for this struct was generated from the following file:

- ratio

6.852 std::ratio_less_equal< _R1, _R2 > Struct Template Reference

`#include <ratio>`
Inheritance diagram for std::ratio_less_equal< _R1, _R2 >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr bool **value**

6.852.1 Detailed Description

template<typename _R1, typename _R2>
struct std::ratio_less_equal< _R1, _R2 >

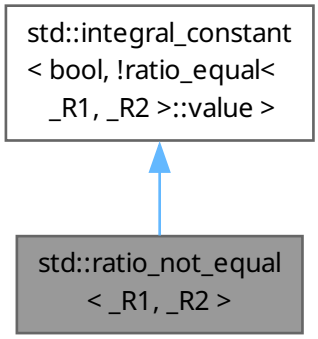
ratio_less_equal
The documentation for this struct was generated from the following file:

- ratio

6.853 std::ratio_not_equal< _R1, _R2 > Struct Template Reference

`#include <ratio>`

Inheritance diagram for std::ratio_not_equal< _R1, _R2 >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr bool **value**

6.853.1 Detailed Description

template<**typename _R1, typename _R2**>
struct std::ratio_not_equal< **_R1, _R2** >

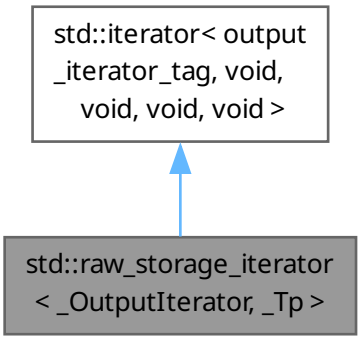
ratio_not_equal
The documentation for this struct was generated from the following file:

- ratio

6.854 std::raw_storage_iterator< _OutputIterator, _Tp > Class Template Reference

`#include <stl_raw_storage_iter.h>`

Inheritance diagram for std::raw_storage_iterator< _OutputIterator, _Tp >:



Public Types

- typedef void difference_type
- typedef output_iterator_tag iterator_category
- typedef void pointer
- typedef void reference
- typedef void value_type

Public Member Functions

- **raw_storage_iterator** (_OutputIterator __x)
- _OutputIterator **base** () const
- raw_storage_iterator & **operator*** ()
- raw_storage_iterator & **operator++** ()
- raw_storage_iterator **operator++** (int)
- raw_storage_iterator & **operator=** (_Tp &&__element)
- raw_storage_iterator & **operator=** (const _Tp &__element)

Protected Attributes

- _OutputIterator **_M_iter**

6.854.1 Detailed Description

template<class _OutputIterator, class _Tp>
class std::raw_storage_iterator< _OutputIterator, _Tp >

This iterator class lets algorithms store their results into uninitialized memory.

6.854.2 Member Typedef Documentation

difference_type

```
typedef void std::iterator< output_iterator_tag, void, void, void, void >::difference_type  [inherited]
```

Distance between iterators is represented as this type.

iterator_category

typedef output_iterator_tag std::iterator< output_iterator_tag, void, void, void, void >::↩
iterator_category [inherited]
One of the tag types.

pointer

typedef void std::iterator< output_iterator_tag, void, void, void, void >::pointer [inherited]
This type represents a pointer-to-value_type.

reference

typedef void std::iterator< output_iterator_tag, void, void, void, void >::reference [inherited]
This type represents a reference-to-value_type.

value_type

typedef void std::iterator< output_iterator_tag, void, void, void, void >::value_type [inherited]
The type "pointed to" by the iterator.
The documentation for this class was generated from the following file:

- stl_raw_storage_iter.h

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

#include <rb_tree>

Public Types

- template<typename _Iter>
  using **__same_value_type**
- typedef std::_Rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc > **_Base**
- typedef _Base::allocator_type **allocator_type**
- typedef _Node_traits::_Const_iterator **const_iterator**
- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef _Node_traits::_Iterator **iterator**
- typedef _Key **key_type**
- typedef value_type * **pointer**
- typedef value_type & **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Val **value_type**

Public Member Functions

- **rb_tree** (const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- bool **__rb_verify** () const
- template<typename _Iterator>
  void **_M_assign_equal** (_Iterator, _Iterator)

- template<typename _Iterator>
  void **_M_assign_unique** (_Iterator, _Iterator)
- template<typename... _Args>
  iterator **_M_emplace_equal** (_Args &&... __args)
- template<typename... _Args>
  iterator **_M_emplace_here** (bool __place_left, _Base_ptr __node, _Args &&... __args)
- template<typename... _Args>
  iterator **_M_emplace_hint_equal** (const_iterator __pos, _Args &&... __args)
- template<typename... _Args>
  iterator **_M_emplace_hint_unique** (const_iterator __pos, _Args &&... __args)
- template<typename... _Args>
  pair< iterator, bool > **_M_emplace_unique** (_Args &&... __args)
- template<typename _Kt>
  size_type **_M_erase_tr** (const _Kt &__x)
- size_type **_M_erase_unique** (const key_type &__x)
- pair< _Base_ptr, _Base_ptr > **_M_get_insert_equal_pos** (const key_type &__k)
- pair< _Base_ptr, _Base_ptr > **_M_get_insert_hint_equal_pos** (const_iterator __pos, const key_type &__k)
- pair< _Base_ptr, _Base_ptr > **_M_get_insert_hint_unique_pos** (const_iterator __pos, const key_type &__k)
- template<typename _Kt>
  pair< _Base_ptr, _Base_ptr > **_M_get_insert_hint_unique_pos_tr** (const_iterator, const _Kt &__k)
- pair< _Base_ptr, _Base_ptr > **_M_get_insert_unique_pos** (const key_type &__k)
- template<typename _Kt>
  pair< _Base_ptr, _Base_ptr > **_M_get_insert_unique_pos_tr** (const _Kt &__k)
- const _Node_allocator & **_M_get_Node_allocator** () const noexcept
- _Node_allocator & **_M_get_Node_allocator** () noexcept
- template<typename _Arg>
  iterator **_M_insert_equal** (_Arg &&__x)
- template<typename _Arg>
  iterator **_M_insert_equal_** (const_iterator __pos, _Arg &&__x)
- template<typename _Arg, typename _NodeGen>
  iterator **_M_insert_equal_** (const_iterator __pos, _Arg &&__x, _NodeGen &)
- template<typename _InputIterator>
  __enable_if_t<!__same_value_type< _InputIterator >::value > **_M_insert_range_equal** (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  __enable_if_t< __same_value_type< _InputIterator >::value > **_M_insert_range_equal** (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  __enable_if_t<!__same_value_type< _InputIterator >::value > **_M_insert_range_unique** (_InputIterator __↩ first, _InputIterator __last)
- template<typename _InputIterator>
  __enable_if_t< __same_value_type< _InputIterator >::value > **_M_insert_range_unique** (_InputIterator __↩ first, _InputIterator __last)
- template<typename _Arg>
  pair< typename _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc >::iterator, bool > **_M_insert_unique** (_Arg &&__v)
- template<typename _Arg>
  pair< iterator, bool > **_M_insert_unique** (_Arg &&__x)
- template<typename _Arg>
  iterator **_M_insert_unique_** (const_iterator __pos, _Arg &&__x)
- template<typename _Arg, typename _NodeGen>
  iterator **_M_insert_unique_** (const_iterator __pos, _Arg &&__x, _NodeGen &)

- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- void **clear** () noexcept
- size_type **count** (const key_type &__k) const
- bool **empty** () const noexcept
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- pair< iterator, iterator > **equal_range** (const key_type &__k)
- pair< const_iterator, const_iterator > **equal_range** (const key_type &__k) const
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __position)
- iterator **erase** (iterator __position)
- iterator **find** (const key_type &__k)
- const_iterator **find** (const key_type &__k) const
- allocator_type **get_allocator** () const noexcept
- _Compare **key_comp** () const
- iterator **lower_bound** (const key_type &__k)
- const_iterator **lower_bound** (const key_type &__k) const
- size_type **max_size** () const noexcept
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rend** () const noexcept
- reverse_iterator **rend** () noexcept
- size_type **size** () const noexcept
- void **swap** (_Rb_tree &__t) noexcept(/∗conditional ∗/)
- iterator **upper_bound** (const key_type &__k)
- const_iterator **upper_bound** (const key_type &__k) const

Protected Types

- typedef _Node_traits::_Base_ptr **_Base_ptr**
- typedef _Node_traits::_Header_t **_Header_t**
- typedef _Node_traits::_Node_ptr **_Node_ptr**

Protected Member Functions

- _Base_ptr **_M_begin** () const noexcept
- _Node_ptr **_M_begin_node** () const noexcept
- template<bool _MoveValue, typename _NodeGen>
  _Node_ptr **_M_clone_node** (_Node_ptr __x, _NodeGen &__node_gen)
- template<typename... _Args>
  void **_M_construct_node** (_Node_ptr __node, _Args &&... __args)
- template<typename... _Args>
  _Node_ptr **_M_create_node** (_Args &&... __args)
- void **_M_destroy_node** (_Node_ptr __p) noexcept
- void **_M_drop_node** (_Node_ptr __p) noexcept
- _Base_ptr **_M_end** () const noexcept
- _Node_ptr **_M_get_node** ()
- template<typename _Key1, typename _Key2>
  bool **_M_key_compare** (const _Key1 &__k1, const _Key2 &__k2) const
- _Base_ptr **_M_leftmost** () const noexcept

- _Base_ptr & **_M_leftmost** () noexcept
- void **_M_put_node** (_Node_ptr __p) noexcept
- _Base_ptr **_M_rightmost** () const noexcept
- _Base_ptr & **_M_rightmost** () noexcept
- _Base_ptr **_M_root** () const noexcept
- _Base_ptr & **_M_root** () noexcept

Static Protected Member Functions

- static const _Key & **_S_key** (_Base_ptr __x)
- static const _Key & **_S_key** (_Node_ptr __x)
- static const _Key & **_S_key** (const _Node &__node)
- static _Base_ptr **_S_left** (_Base_ptr __x) noexcept
- static _Node_ptr **_S_left** (_Node_ptr __x)
- static _Base_ptr **_S_right** (_Base_ptr __x) noexcept
- static _Node_ptr **_S_right** (_Node_ptr __x) noexcept

Protected Attributes

- _Rb_tree_impl< _Compare > **_M_impl**

6.855.1 Detailed Description

template<class _Key, class _Value, class _KeyOfValue, class _Compare, class _Alloc = std::allocator<_↩ Value>>
struct __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >

This is an SGI extension.

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

The documentation for this struct was generated from the following file:

- rb_tree

6.856 __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference

`#include <rb_tree_.hpp>`

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef base_type::const_reverse_iterator **const_reverse_iterator**
- typedef rb_tree_tag **container_category**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::key_const_pointer **key_const_pointer**

- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_pointer **key_pointer**
- typedef base_type::key_reference **key_reference**
- typedef base_type::key_type **key_type**
- typedef base_type::mapped_const_pointer **mapped_const_pointer**
- typedef base_type::mapped_const_reference **mapped_const_reference**
- typedef base_type::mapped_pointer **mapped_pointer**
- typedef base_type::mapped_reference **mapped_reference**
- typedef base_type::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- typedef base_type::node_update **node_update**
- typedef base_type::const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef base_type::reverse_iterator **reverse_iterator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef base_type::value_type **value_type**

Public Member Functions

- **rb_tree_map** (const Cmp_Fn &)
- **rb_tree_map** (const Cmp_Fn &, const node_update &)
- **rb_tree_map** (const rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It>
  void **copy_from_range** (It, It)
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator)
- bool **erase** (key_const_reference)
- reverse_iterator **erase** (reverse_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- std::pair< point_iterator, bool > **insert** (const_reference)
- void **join** (rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **lower_bound** (key_const_reference)
- point_const_iterator **lower_bound** (key_const_reference) const
- size_type **max_size** () const
- node_iterator node_begin ()

- node_const_iterator node_begin () const
- node_iterator node_end ()
- node_const_iterator node_end () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- size_type **size** () const
- void **split** (key_const_reference, rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference)
- point_const_iterator **upper_bound** (key_const_reference) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Protected Types

- typedef node_alloc_traits::value_type **node**
- typedef node_alloc_traits::allocator_type **node_allocator**
- typedef traits_type::null_node_update_pointer **null_node_update_pointer**
- typedef types_traits< Key, Mapped, _Alloc, false > **traits_base**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- template<typename Node_Update_>
  void **apply_update** (node_pointer, Node_Update_ *)
- void **apply_update** (node_pointer, null_node_update_pointer)
- std::pair< node_pointer, bool > **erase** (node_pointer)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, false_type)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, true_type)
- void **initialize_min_max** ()
- iterator **insert_imp_empty** (const_reference)
- std::pair< point_iterator, bool > **insert_leaf** (const_reference)
- iterator **insert_leaf_new** (const_reference, node_pointer, bool)
- void **join_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **join_prep** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- size_type **recursive_count** (node_pointer) const
- void **rotate_left** (node_pointer)
- void **rotate_parent** (node_pointer)
- void **rotate_right** (node_pointer)
- void **split_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **split_prep** (key_const_reference, bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _↩
  Alloc > &)
- void **update_min_max_for_erased_node** (node_pointer)
- void **update_subtree_size** (node_pointer)
- template<typename Node_Update_>
  void **update_to_top** (node_pointer, Node_Update_ *)
- void **update_to_top** (node_pointer, null_node_update_pointer)
- void **value_swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)

Static Protected Member Functions

- static void **clear_imp** (node_pointer)

Protected Attributes

- node_pointer **m_p_head**
- size_type **m_size**

Static Protected Attributes

- static node_allocator **s_node_allocator**

6.856.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Red-Black tree.
This implementation uses an idea from the SGI STL (using a *header* node which is needed for efficient iteration).

6.856.2 Member Function Documentation

node_begin() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin () [inline], [inherited]
```

Returns a node_iterator corresponding to the node at the root of the tree.

node_begin() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin () const [inline], [inherited]
```

Returns a const node_iterator corresponding to the node at the root of the tree.

node_end() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end () [inline], [inherited]
```

Returns a node_iterator corresponding to a node just after a leaf of the tree.

node_end() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
```

bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↩ gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↩ end () const [inline], [inherited]

Returns a const node_iterator corresponding to a node just after a leaf of the tree.

The documentation for this class was generated from the following file:

- rb_tree_.hpp

6.857 __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference

#include <node.hpp>

Public Types

- typedef rebind_traits< _Alloc, metadata_type >::const_reference **metadata_const_reference**
- typedef rebind_traits< _Alloc, metadata_type >::reference **metadata_reference**
- typedef Metadata **metadata_type**
- typedef rebind_traits< _Alloc, rb_tree_node_ >::pointer **node_pointer**
- typedef rebind_traits< _Alloc, rb_tree_node_ >::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- metadata_reference **get_metadata** ()
- metadata_const_reference **get_metadata** () const
- bool **special** () const

Public Attributes

- metadata_type **m_metadata**
- node_pointer **m_p_left**
- node_pointer **m_p_parent**
- node_pointer **m_p_right**
- bool **m_red**
- size_type **m_subtree_size**
- value_type **m_value**

6.857.1 Detailed Description

template<typename Value_Type, class Metadata, typename _Alloc>
struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >

Node for Red-Black trees.

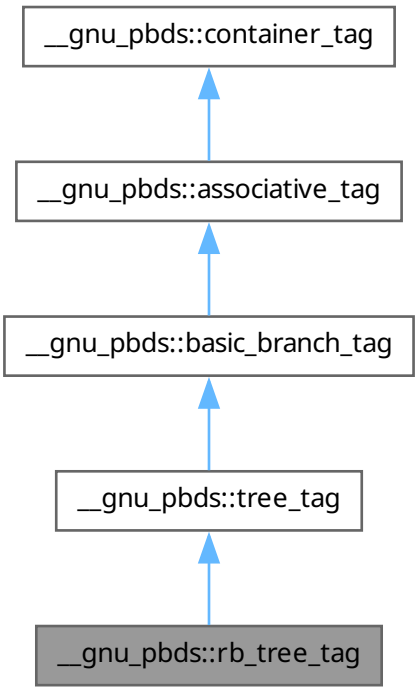
The documentation for this struct was generated from the following file:

- rb_tree_map_/node.hpp

6.858 __gnu_pbds::rb_tree_tag Struct Reference

#include <tag_and_trait.hpp>

Inheritance diagram for __gnu_pbds::rb_tree_tag:



6.858.1 Detailed Description

Red-black tree.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.859 __gnu_pbds::detail::rc< _Node, _Alloc > Class Template Reference

`#include <rc.hpp>`

Public Types

- typedef entry_const_pointer **const_iterator**
- typedef node_pointer **entry**

Public Member Functions

- **rc** (const rc &)
- const const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- const const_iterator **end** () const

- void **pop** ()
- void **push** (entry)
- size_type **size** () const
- void **swap** (rc &)
- node_pointer **top** () const

6.859.1 Detailed Description

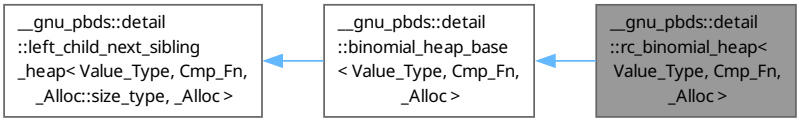
template<typename _Node, typename _Alloc>
class __gnu_pbds::detail::rc< _Node, _Alloc >

Redundant binary counter.
The documentation for this class was generated from the following file:

- rc.hpp

6.860 __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

`#include <rc_binomial_heap_.hpp>`
Inheritance diagram for __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >:



Public Types

- typedef base_type::allocator_type **allocator_type**
- typedef base_type::cmp_fn **cmp_fn**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- **rc_binomial_heap** (const Cmp_Fn &)
- **rc_binomial_heap** (const rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > &)
- iterator **begin** ()
- void **clear** ()

- bool **empty** () const
- iterator **end** ()
- void **erase** (point_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- void **join** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **join** (rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred>
  void **split** (Pred, binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- template<typename Pred>
  void **split** (Pred, rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- void **swap** (rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef alloc_traits::allocator_type **node_allocator**
- typedef _Alloc::size_type **node_metadata**
- typedef std::pair< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It>
  void **copy_from_range** (It, It)
- void **find_max** ()
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)
- void **swap** (binomial_heap_base< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static void **make_child_of** (node_pointer, node_pointer)
- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_max**
- node_pointer **m_p_root**
- size_type **m_size**

6.860.1 Detailed Description

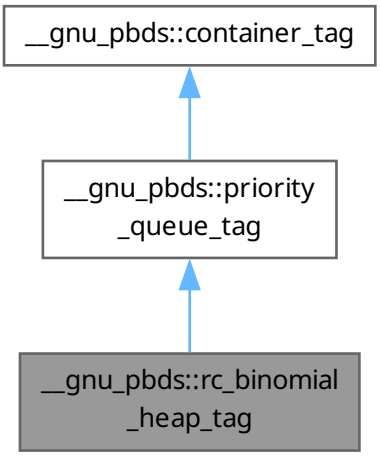
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >

Redundant-counter binomial heap.
The documentation for this class was generated from the following file:

- rc_binomial_heap_.hpp

6.861 __gnu_pbds::rc_binomial_heap_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::rc_binomial_heap_tag:



6.861.1 Detailed Description

Redundant-counter binomial-heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.862 std::simd::rebind< _Tp, _Mp, _Traits > Struct Template Reference

`#include <simd_mask.h>`

Public Types

- using **type**

6.862.1 Detailed Description

template<__vectorizable _Tp, __simd_mask_type _Mp, _ArchTraits _Traits>
struct std::simd::rebind< _Tp, _Mp, _Traits >

As above, except for `basic_mask`.
The documentation for this struct was generated from the following file:

- simd_mask.h

6.863 std::simd::rebind< _Tp, _Vp, _Traits > Struct Template Reference

`#include <simd_mask.h>`

Public Types

- using **type**

6.863.1 Detailed Description

template<__vectorizable _Tp, __simd_vec_type _Vp, _ArchTraits _Traits>
struct std::simd::rebind< _Tp, _Vp, _Traits >

Computes a member `type basic_vec<_Tp, Abi>`, where `Abi` is chosen such that the number of elements is equal to `_Vp::size()` and features of the ABI tag (such as the internal representation of masks, or storage order of complex components) are preserved.
The documentation for this struct was generated from the following file:

- simd_mask.h

6.864 __gnu_pbds::detail::rebind_traits< _Alloc, T > Struct Template Reference

`#include <types_traits.hpp>`

Public Types

- using **const_reference**
- using **reference**

6.864.1 Detailed Description

template<typename _Alloc, typename T>
struct __gnu_pbds::detail::rebind_traits< _Alloc, T >

Consistent API for accessing allocator-related types.
The documentation for this struct was generated from the following file:

- types_traits.hpp

6.865 std::filesystem::recursive_directory_iterator Class Reference

`#include <filesystem>`

Public Types

- typedef ptrdiff_t **difference_type**
- typedef input_iterator_tag **iterator_category**
- typedef const directory_entry * **pointer**
- typedef const directory_entry & **reference**
- typedef directory_entry **value_type**

Public Member Functions

- **recursive_directory_iterator** (const path &__p)
- **recursive_directory_iterator** (const path &__p, directory_options __options)
- **recursive_directory_iterator** (const path &__p, directory_options __options, error_code &__ec)
- **recursive_directory_iterator** (const path &__p, error_code &__ec)
- **recursive_directory_iterator** (const recursive_directory_iterator &)=default
- **recursive_directory_iterator** (recursive_directory_iterator &&)=default
- int **depth** () const noexcept
- void **disable_recursion_pending** () noexcept
- recursive_directory_iterator & **increment** (error_code &__ec)
- const directory_entry & **operator*** () const noexcept
- recursive_directory_iterator & **operator++** ()
- __directory_iterator_proxy **operator++** (int)
- const directory_entry * **operator->** () const noexcept
- recursive_directory_iterator & **operator=** (const recursive_directory_iterator &__rhs) noexcept
- recursive_directory_iterator & **operator=** (recursive_directory_iterator &&__rhs) noexcept
- bool **operator==** (default_sentinel_t) const noexcept
- directory_options **options** () const noexcept
- void **pop** ()
- void **pop** (error_code &)
- bool **recursion_pending** () const noexcept

Friends

- uintmax_t **filesystem::remove_all** (const path &)
- uintmax_t **filesystem::remove_all** (const path &, error_code &)
- bool **operator==** (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs) noexcept

6.865.1 Detailed Description

Iterator type for recursively traversing a directory hierarchy.

Since

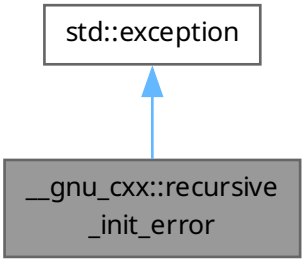
C++17

The documentation for this class was generated from the following file:

- bits/fs_dir.h

6.866 __gnu_cxx::recursive_init_error Class Reference

`#include <cxxabi.h>`
Inheritance diagram for __gnu_cxx::recursive_init_error:



Public Member Functions

- virtual constexpr const char * what () const noexcept

6.866.1 Detailed Description

Exception thrown by __cxa_guard_acquire.
C++ 2011 6.7 [stmt.dcl]/4: If control re-enters the declaration recursively while the variable is being initialized, the behavior is undefined.
Since we already have a library function to handle locking, we might as well check for this situation and throw an exception. We use the second byte of the guard variable to remember that we're in the middle of an initialization.

6.866.2 Member Function Documentation

what()

`virtual constexpr const char * std::exception::what () const [inline], [constexpr], [virtual], [noexcept], [inherited]`
Returns a C-style character string describing the general cause of the current error.
Reimplemented in std::bad_alloc, std::bad_cast, std::bad_exception, std::bad_function_call, std::bad_typeid, std::bad_weak_ptr, std::experimental::filesystem::v1::filesystem_error, std::experimental::fundamentals_v1::bad_any_cast, std::filesystem::filesystem_error, std::future_error, std::ios_base::failure, std::logic_error, and std::runtime_error.
The documentation for this class was generated from the following file:

- cxxabi.h

6.867 std::recursive_mutex Class Reference

`#include <mutex>`

Public Types

- typedef __native_type * **native_handle_type**

Public Member Functions

- **recursive_mutex** (const recursive_mutex &)=delete
- void **lock** ()
- native_handle_type **native_handle** () noexcept
- recursive_mutex & **operator=** (const recursive_mutex &)=delete
- bool **try_lock** () noexcept
- void **unlock** ()

6.867.1 Detailed Description

The standard recursive mutex type.
A recursive mutex can be locked more than once by the same thread. Other threads cannot lock the mutex until the owning thread unlocks it as many times as it was locked.

Since

C++11

The documentation for this class was generated from the following file:

- mutex

6.868 std::recursive_timed_mutex Class Reference

`#include <mutex>`

Public Types

- typedef __native_type * **native_handle_type**

Public Member Functions

- **recursive_timed_mutex** (const recursive_timed_mutex &)=delete
- void **lock** ()
- native_handle_type **native_handle** () noexcept
- recursive_timed_mutex & **operator=** (const recursive_timed_mutex &)=delete
- bool **try_lock** () noexcept
- template<class _Rep, class _Period>
  bool **try_lock_for** (const chrono::duration< _Rep, _Period > &__rtime)
- template<class _Clock, class _Duration>
  bool **try_lock_until** (const chrono::time_point< _Clock, _Duration > &__atime)
- void **unlock** ()

Friends

- class **__timed_mutex_impl**< **recursive_timed_mutex** >

6.868.1 Detailed Description

The standard recursive timed mutex type.
A recursive mutex that supports a timeout when trying to acquire the lock. A recursive mutex can be locked more than once by the same thread. Other threads cannot lock the mutex until the owning thread unlocks it as many times as it was locked.

Since

C++11

The documentation for this class was generated from the following file:

- mutex

6.869 std::bitset< _Nb >::reference Class Reference

`#include <bitset>`

Public Member Functions

- **reference** (const reference &)=default
- constexpr reference & **flip** () noexcept
- constexpr **operator bool** () const noexcept
- constexpr const reference & **operator=** (bool __x) const noexcept
- constexpr reference & **operator=** (bool __x) noexcept
- constexpr reference & **operator=** (const reference &__j) noexcept
- constexpr bool **operator~** () const noexcept

Friends

- class **bitset**
- constexpr friend void **swap** (bool &__x, reference __y) noexcept
- constexpr friend void **swap** (reference __x, bool &__y) noexcept
- constexpr friend void **swap** (reference __x, reference __y) noexcept

6.869.1 Detailed Description

template<size_t _Nb>
class std::bitset< _Nb >::reference

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.
Most users will never need to use this class directly; conversions to and from bool are automatic and should be transparent. Overloaded operators help to preserve the illusion.
(On a typical system, this *bit reference* is 64 times the size of an actual bit. Ha.)
The documentation for this class was generated from the following file:

- bitset

6.870 std::tr2::dynamic_bitset< _WordT, _Alloc >::reference Class Reference

`#include <dynamic_bitset>`

Public Member Functions

- **reference** (dynamic_bitset &__b, size_type __pos) noexcept
- reference & **flip** () noexcept
- **operator bool** () const noexcept
- reference & **operator=** (bool __x) noexcept
- reference & **operator=** (const reference &__j) noexcept
- bool **operator~** () const noexcept

Friends

- class **dynamic_bitset**

6.870.1 Detailed Description

template<typename _WordT = unsigned long long, typename _Alloc = std::allocator<_WordT>>
class std::tr2::dynamic_bitset< _WordT, _Alloc >::reference

This encapsulates the concept of a single bit. An instance of this class is a proxy for an actual bit; this way the individual bit operations are done as faster word-size bitwise instructions.
Most users will never need to use this class directly; conversions to and from bool are automatic and should be transparent. Overloaded operators help to preserve the illusion.
(On a typical system, this "bit %reference" is 64 times the size of an actual bit. Ha.)
The documentation for this class was generated from the following file:

- dynamic_bitset

6.871 std::reference_wrapper< _Tp > Class Template Reference

`#include <refwrap.h>`

Public Types

- typedef _Tp **type**

Public Member Functions

- template<typename _Up, typename = __not_same<_Up>, typename = decltype(reference_wrapper::_S_fun(std::declval<_Up>()))>
  constexpr **reference_wrapper** (_Up &&__uref) noexcept(noexcept(reference_wrapper::_S_fun(std::declval< ↩
  _Up >())))
- **reference_wrapper** (const reference_wrapper &)=default
- constexpr _Tp & **get** () const noexcept
- constexpr **operator _Tp &** () const noexcept
- template<typename... _Args>
  constexpr __invoke_result< _Tp &, _Args... >::type **operator()** (_Args &&... __args) const noexcept(__is_↩
  nothrow_invocable< _Tp &, _Args... >::value)
- reference_wrapper & **operator=** (const reference_wrapper &)=default

Friends

- constexpr auto **operator<=>** (reference_wrapper __x, const _Tp &__y)
- constexpr auto **operator<=>** (reference_wrapper __x, reference_wrapper __y)
- constexpr auto **operator<=>** (reference_wrapper __x, reference_wrapper< const _Tp > __y)
- constexpr bool **operator==** (reference_wrapper __x, const _Tp &__y)
- constexpr bool **operator==** (reference_wrapper __x, reference_wrapper __y)
- constexpr bool **operator==** (reference_wrapper __x, reference_wrapper< const _Tp > __y)

Related Symbols

(Note that these are not member symbols.)

- template<typename _Tp>
  constexpr reference_wrapper< _Tp > ref (_Tp &__t) noexcept
- template<typename _Tp>
  constexpr reference_wrapper< const _Tp > cref (const _Tp &__t) noexcept

- template<typename _Tp>
  constexpr reference_wrapper< _Tp > ref (reference_wrapper< _Tp > __t) noexcept
- template<typename _Tp>
  constexpr reference_wrapper< const _Tp > cref (reference_wrapper< _Tp > __t) noexcept

6.871.1 Detailed Description

template<typename _Tp>
class std::reference_wrapper< _Tp >

Primary class template for reference_wrapper.

6.871.2 Friends And Related Symbol Documentation

cref() [1/2]

```
template<typename _Tp>
reference_wrapper< const _Tp > cref (
            const _Tp & __t)  [related]
```

Denotes a const reference should be taken to a variable.

cref() [2/2]

```
template<typename _Tp>
reference_wrapper< const _Tp > cref (
            reference_wrapper< _Tp > __t)  [related]
```

std::cref overload to prevent wrapping a reference_wrapper

ref() [1/2]

```
template<typename _Tp>
reference_wrapper< _Tp > ref (
            _Tp & __t)  [related]
```

Denotes a reference should be taken to a variable.

ref() [2/2]

```
template<typename _Tp>
reference_wrapper< _Tp > ref (
            reference_wrapper< _Tp > __t)  [related]
```

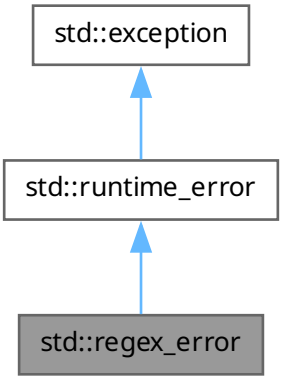
std::ref overload to prevent wrapping a reference_wrapper
The documentation for this class was generated from the following files:

- type_traits
- refwrap.h

6.872 std::regex_error Class Reference

```
#include <regex>
```

Inheritance diagram for std::regex_error:



Public Member Functions

- regex_error (error_type __ecode)
- regex_constants::error_type code () const noexcept
- virtual const char * what () const noexcept

6.872.1 Detailed Description

A regular expression exception class.
The regular expression library throws objects of this class on error.

Since

C++11

6.872.2 Constructor & Destructor Documentation

regex_error()

```
std::regex_error::regex_error (
            error_type __ecode)  [explicit]
```

Constructs a regex_error object.

Parameters

__ecode	the regex error code.

References regex_error().
Referenced by regex_error().

6.872.3 Member Function Documentation

code()

```
regex_constants::error_type std::regex_error::code () const  [inline], [noexcept]
```

Gets the regex error code.

Returns

the regex error code.

References code().
Referenced by code().

what()

```
virtual const char * std::runtime_error::what () const  [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::experimental::filesystem::v1::filesystem_error, std::filesystem::filesystem_error, and std::ios_base::failure.
References what().
Referenced by what().
The documentation for this class was generated from the following file:

- regex_error.h

6.873 std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference

```
#include <regex>
```

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef std::input_iterator_tag **iterator_concept**
- typedef const value_type * **pointer**
- typedef const value_type & **reference**
- typedef basic_regex< _Ch_type, _Rx_traits > **regex_type**
- typedef match_results< _Bi_iter > **value_type**

Public Member Functions

- regex_iterator ()=default
- regex_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, regex_constants::match_flag_type __m=regex_constants::match_default)
- **regex_iterator** (_Bi_iter, _Bi_iter, const regex_type &&, regex_constants::match_flag_type=regex_constants::match_default)=delete
- regex_iterator (const regex_iterator &)=default
- const value_type & operator* () const noexcept
- regex_iterator & operator++ ()
- regex_iterator operator++ (int)
- const value_type * operator-> () const noexcept
- regex_iterator & operator= (const regex_iterator &)=default
- bool operator== (const regex_iterator &) const noexcept
- bool **operator==** (default_sentinel_t) const noexcept

6.873.1 Detailed Description

template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >

An iterator adaptor that will provide repeated calls of regex_search over a range until no more matches remain.

Since

C++11

6.873.2 Constructor & Destructor Documentation

regex_iterator() [1/3]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator ()  [default]
```

Provides a singular iterator, useful for indicating one-past-the-end of a range.
Referenced by std::regex_iterator< const char * >::regex_iterator(), and operator==().

regex_iterator() [2/3]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (
             _Bi_iter __a,
             _Bi_iter __b,
             const regex_type & __re,
             regex_constants::match_flag_type __m = regex_constants::match_default)  [inline]
```

Constructs a regex_iterator...

Parameters

_a	[IN] The start of a text range to search.
_b	[IN] One-past-the-end of the text range to search.
_re	[IN] The regular expression to match.
_m	[IN] Policy flags for match rules.

regex_iterator() [3/3]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_iterator (
             const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & )  [default]
```

Copy constructs a regex_iterator.

6.873.3 Member Function Documentation

operator*()

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type & std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* () const
[inline], [noexcept]
```

Dereferences a regex_iterator.

operator++() [1/2]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_iterator< _Bi_iter, _Ch_type, _Rx↩
_traits >::operator++ ()
```

Increments a regex_iterator.

References std::regex_constants::match_continuous, std::regex_constants::match_not_null, std::regex_constants::match_prev_avail, and std::regex_search().

operator++() [2/2]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_iterator std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (
            int )  [inline]
```

Postincrements a regex_iterator.

operator->()

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type * std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> () const
[inline], [noexcept]
```

Selects a regex_iterator member.

operator=()

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_iterator & std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator= (
            const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & )  [default]
```

Copy assigns one regex_iterator to another.

operator==()

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== (
            const regex_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) const  [noexcept]
```

Tests the equivalence of two regex iterators.

References regex_iterator().

The documentation for this class was generated from the following files:

- regex.h
- regex.tcc

6.874 std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > Class Template Reference

`#include <regex>`

Public Types

- typedef std::ptrdiff_t **difference_type**
- typedef std::forward_iterator_tag **iterator_category**
- typedef std::input_iterator_tag **iterator_concept**
- typedef const value_type * **pointer**
- typedef const value_type & **reference**
- typedef basic_regex< _Ch_type, _Rx_traits > **regex_type**
- typedef sub_match< _Bi_iter > **value_type**

Public Member Functions

- regex_token_iterator ()
- template<std::size_t _Nm>
  regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const int(&__submatches)[_Nm], regex_constants::match_flag_type __m=regex_constants::match_default)
- regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, const std::vector< int > &__submatches, regex_constants::match_flag_type __m=regex_constants::match_default)
- regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, initializer_list< int > __submatches, regex_constants::match_flag_type __m=regex_constants::match_default)
- regex_token_iterator (_Bi_iter __a, _Bi_iter __b, const regex_type &__re, int __submatch=0, regex_constants::match_flag_type __m=regex_constants::match_default)
- template<std::size_t _Nm>
  regex_token_iterator (_Bi_iter, _Bi_iter, const regex_type &&, const int(&)[_Nm], regex_constants::match_flag_type=regex_consta
- **regex_token_iterator** (_Bi_iter, _Bi_iter, const regex_type &&, const std::vector< int > &, regex_constants::match_flag_type=regex
- **regex_token_iterator** (_Bi_iter, _Bi_iter, const regex_type &&, initializer_list< int >, regex_constants::match_flag_type=regex_cons
- **regex_token_iterator** (_Bi_iter, _Bi_iter, const regex_type &&, int=0, regex_constants::match_flag_type=regex_constants::match_c
- regex_token_iterator (const regex_token_iterator &__rhs)
- const value_type & operator* () const
- regex_token_iterator & operator++ ()
- regex_token_iterator operator++ (int)
- const value_type * operator-> () const
- regex_token_iterator & operator= (const regex_token_iterator &__rhs)
- bool operator== (const regex_token_iterator &__rhs) const
- bool **operator==** (default_sentinel_t) const noexcept

6.874.1 Detailed Description

template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type, typename _Rx_traits = regex_traits<_Ch_type>>
class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >

Iterates over submatches in a range (or *splits* a text string).
The purpose of this iterator is to enumerate all, or all specified, matches of a regular expression within a text range. The dereferenced value of an iterator of this class is a std::sub_match object.

Since

C++11

6.874.2 Constructor & Destructor Documentation

regex_token_iterator() [1/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator ()  [inline]
```

Default constructs a regex_token_iterator.

A default-constructed regex_token_iterator is a singular iterator that will compare equal to the one-past-the-end value for any iterator of the same type.

Referenced by operator++(), operator=(), and operator==().

regex_token_iterator() [2/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
            _Bi_iter __a,
            _Bi_iter __b,
            const regex_type & __re,
            int __submatch = 0,
            regex_constants::match_flag_type __m = regex_constants::match_default)  [inline]
```

Constructs a regex_token_iterator...

Parameters

__a	[IN] The start of the text to search.
__b	[IN] One-past-the-end of the text to search.
__re	[IN] The regular expression to search for.
__submatch	[IN] Which submatch to return. There are some special values for this parameter: • -1 each enumerated subexpression does NOT match the regular expression (aka field splitting) • 0 the entire string matching the subexpression is returned for each match within the text. • >0 enumerates only the indicated subexpression from a match within the text.
__m	[IN] Policy flags for match rules.

regex_token_iterator() [3/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
            _Bi_iter __a,
            _Bi_iter __b,
            const regex_type & __re,
            const std::vector< int > & __submatches,
            regex_constants::match_flag_type __m = regex_constants::match_default)  [inline]
```

Constructs a regex_token_iterator...

Parameters

__a	[IN] The start of the text to search.
__b	[IN] One-past-the-end of the text to search.
__re	[IN] The regular expression to search for.
__submatches	[IN] A list of subexpressions to return for each regular expression match within the text.
__m	[IN] Policy flags for match rules.

regex_token_iterator() [4/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
            _Bi_iter __a,
            _Bi_iter __b,
            const regex_type & __re,
            initializer_list< int > __submatches,
            regex_constants::match_flag_type __m = regex_constants::match_default)  [inline]
```

Constructs a regex_token_iterator...

Parameters

__a	[IN] The start of the text to search.
__b	[IN] One-past-the-end of the text to search.
__re	[IN] The regular expression to search for.
__submatches	[IN] A list of subexpressions to return for each regular expression match within the text.
__m	[IN] Policy flags for match rules.

regex_token_iterator() [5/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
template<std::size_t _Nm>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
            _Bi_iter __a,
            _Bi_iter __b,
            const regex_type & __re,
            const int(&) __submatches[_Nm],
            regex_constants::match_flag_type __m = regex_constants::match_default)  [inline]
```

Constructs a regex_token_iterator...

Parameters

__a	[IN] The start of the text to search.
__b	[IN] One-past-the-end of the text to search.
__re	[IN] The regular expression to search for.
__submatches	[IN] A list of subexpressions to return for each regular expression match within the text.
__m	[IN] Policy flags for match rules.

regex_token_iterator() [6/6]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::regex_token_iterator (
            const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs)  [inline]
```

Copy constructs a regex_token_iterator.

Parameters

__rhs	[IN] A regex_token_iterator to copy.

6.874.3 Member Function Documentation

operator*()

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type & std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator* ()
const  [inline]
```

Dereferences a regex_token_iterator.

operator++() [1/2]

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_token_iterator< _Bi_iter, _↩
Ch_type, _Rx_traits >::operator++ ()
```

Increments a regex_token_iterator.
References regex_token_iterator().

operator++() [2/2]

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
regex_token_iterator std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator++ (
            int )  [inline]
```

Postincrements a regex_token_iterator.

operator->()

```
template<typename _Bi_iter, typename _Ch_type = typename iterator_traits<_Bi_iter>::value_type,
typename _Rx_traits = regex_traits<_Ch_type>>
const value_type * std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator-> ()
const  [inline]
```

Selects a regex_token_iterator member.

operator=()

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & std::regex_token_iterator< _Bi_iter, _↩
Ch_type, _Rx_traits >::operator= (
            const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs)
```

Assigns a regex_token_iterator to another.

Parameters

__rhs	[IN] A regex_token_iterator to copy.

References regex_token_iterator().

operator==()

```
template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
bool std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >::operator== (
            const regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits > & __rhs) const
```

Compares a regex_token_iterator to another for equality.
References regex_token_iterator().
The documentation for this class was generated from the following files:

- regex.h
- regex.tcc

6.875 std::regex_traits< _Ch_type > Class Template Reference

```
#include <regex>
```

Public Types

- typedef _RegexMask **char_class_type**
- typedef _Ch_type **char_type**
- typedef std::locale **locale_type**
- typedef std::basic_string< char_type > **string_type**

Public Member Functions

- regex_traits ()
- locale_type getloc () const
- locale_type imbue (locale_type __loc)
- bool isctype (_Ch_type __c, char_class_type __f) const
- template<typename _Fwd_iter>
  char_class_type lookup_classname (_Fwd_iter __first, _Fwd_iter __last, bool __icase=false) const
- template<typename _Fwd_iter>
  string_type lookup_collatename (_Fwd_iter __first, _Fwd_iter __last) const

- template<typename _Fwd_iter>
  string_type transform (_Fwd_iter __first, _Fwd_iter __last) const
- template<typename _Fwd_iter>
  string_type transform_primary (_Fwd_iter __first, _Fwd_iter __last) const
- char_type translate (char_type __c) const
- char_type translate_nocase (char_type __c) const
- int value (_Ch_type __ch, int __radix) const

Static Public Member Functions

- static std::size_t length (const char_type *__p)

Protected Attributes

- locale_type **_M_locale**

6.875.1 Detailed Description

template<typename _Ch_type>
class std::regex_traits< _Ch_type >

Describes aspects of a regular expression.
A regular expression traits class that satisfies the requirements of section [28.7].
The class regex is parameterized around a set of related types and functions used to complete the definition of its semantics. This class satisfies the requirements of such a traits class.

Since

C++11

6.875.2 Constructor & Destructor Documentation

regex_traits()

```
template<typename _Ch_type>
std::regex_traits< _Ch_type >::regex_traits ()  [inline]
```

Constructs a default traits object.
Referenced by value().

6.875.3 Member Function Documentation

getloc()

```
template<typename _Ch_type>
locale_type std::regex_traits< _Ch_type >::getloc () const  [inline]
```

Gets a copy of the current locale in use by the regex_traits object.

imbue()

```
template<typename _Ch_type>
locale_type std::regex_traits< _Ch_type >::imbue (
            locale_type __loc)  [inline]
```

Imbues the regex_traits object with a copy of a new locale.

Parameters

__loc	A locale.

Returns

a copy of the previous locale in use by the regex_traits object.

Note

Calling imbue with a different locale than the one currently in use invalidates all cached data held by ∗this.

isctype()

```
template<typename _Ch_type>
bool std::regex_traits< _Ch_type >::isctype (
             _Ch_type __c,
             char_class_type __f) const
```

Determines if c is a member of an identified class.

Parameters

_↩ _c	a character.
_↩ _f	a class type (as returned from lookup_classname).

Returns

true if the character __c is a member of the classification represented by __f, false otherwise.

Exceptions

std::bad_cast	if the current locale does not have a ctype facet.

References std::use_facet().

length()

```
template<typename _Ch_type>
std::size_t std::regex_traits< _Ch_type >::length (
             const char_type * __p)  [inline], [static]
```

Gives the length of a C-style string starting at __p.

Parameters

_↩ _p	a pointer to the start of a character sequence.

Returns

the number of characters between *__p and the first default-initialized value of type char_type. In other words, uses the C-string algorithm for determining the length of a sequence of characters.

lookup_classname()

```
template<typename _Ch_type>
template<typename _Fwd_iter>
regex_traits< _Ch_type >::char_class_type std::regex_traits< _Ch_type >::lookup_classname (
             _Fwd_iter __first,
             _Fwd_iter __last,
             bool __icase = false) const
```

Maps one or more characters to a named character classification.

Parameters

__first	beginning of the character sequence.
__last	one-past-the-end of the character sequence.
__icase	ignores the case of the classification name.

Returns

an unspecified value that represents the character classification named by the character sequence designated by the iterator range [__first, __last). If icase is true, the returned mask identifies the classification regardless of the case of the characters to be matched (for example, [[:lower:]] is the same as [[:alpha:]]), otherwise a case-dependent classification is returned. The value returned shall be independent of the case of the characters in the character sequence. If the name is not recognized then returns a value that compares equal to 0.

At least the following names (or their wide-character equivalent) are supported.

- d
- w
- s
- alnum
- alpha
- blank
- cntrl
- digit
- graph
- lower

- print
- punct
- space
- upper
- xdigit

References std::use_facet().

lookup_collatename()

```
template<typename _Ch_type>
template<typename _Fwd_iter>
regex_traits< _Ch_type >::string_type std::regex_traits< _Ch_type >::lookup_collatename (
            _Fwd_iter __first,
            _Fwd_iter __last) const
```

Gets a collation element by name.

Parameters

__first	beginning of the collation element name.
__last	one-past-the-end of the collation element name.

Returns

a sequence of one or more characters that represents the collating element consisting of the character sequence designated by the iterator range [__first, __last). Returns an empty string if the character sequence is not a valid collating element.

References std::basic_string< _CharT, _Traits, _Alloc >::empty(), and std::use_facet().

transform()

```
template<typename _Ch_type>
template<typename _Fwd_iter>
string_type std::regex_traits< _Ch_type >::transform (
            _Fwd_iter __first,
            _Fwd_iter __last) const  [inline]
```

Gets a sort key for a character sequence.

Parameters

__first	beginning of the character sequence.
__last	one-past-the-end of the character sequence.

Returns a sort key for the character sequence designated by the iterator range [F1, F2) such that if the character sequence [G1, G2) sorts before the character sequence [H1, H2) then v.transform(G1, G2) < v.transform(H1, H2).
What this really does is provide a more efficient way to compare a string to multiple other strings in locales with fancy collation rules and equivalence classes.

Returns

a locale-specific sort key equivalent to the input range.

Exceptions

std::bad_cast	if the current locale does not have a collate facet.

transform_primary()

```
template<typename _Ch_type>
template<typename _Fwd_iter>
string_type std::regex_traits< _Ch_type >::transform_primary (
            _Fwd_iter __first,
            _Fwd_iter __last) const  [inline]
```

Gets a sort key for a character sequence, independent of case.

Parameters

__first	beginning of the character sequence.
__last	one-past-the-end of the character sequence.

Effects: if `typeid(use_facet<collate<_Ch_type>>(getloc())) == typeid(collate_byname<_Ch_type>)` and the form of the sort key returned by `collate_byname<_Ch_type>::transform(__first, __last)` is known and can be converted into a primary sort key then returns that key, otherwise returns an empty string.

Todo Implement this function correctly.

translate()

```
template<typename _Ch_type>
char_type std::regex_traits< _Ch_type >::translate (
            char_type __c) const  [inline]
```

Performs the identity translation.

Parameters

__c	A character to the locale-specific character set.

Returns

__c.

translate_nocase()

```
template<typename _Ch_type>
char_type std::regex_traits< _Ch_type >::translate_nocase (
            char_type __c) const  [inline]
```

Translates a character into a case-insensitive equivalent.

Parameters

_↩ _c	A character to the locale-specific character set.

Returns

the locale-specific lower-case equivalent of __c.

Exceptions

std::bad_cast	if the imbued locale does not support the ctype facet.

value()

```
template<typename _Ch_type>
int std::regex_traits< _Ch_type >::value (
             _Ch_type __ch,
             int __radix) const
```

Converts a digit to an int.

Parameters

__ch	a character representing a digit.
__radix	the radix if the numeric conversion (limited to 8, 10, or 16).

Returns

the value represented by the digit __ch in base radix if the character __ch is a valid digit in base radix; otherwise returns -1.

References regex_traits(), and std::use_facet().
The documentation for this class was generated from the following files:

- regex.h
- regex.tcc

6.876 std::remove_all_extents< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.876.1 Detailed Description

template<typename _Tp>
struct std::remove_all_extents< _Tp >

remove_all_extents
The documentation for this struct was generated from the following file:

- type_traits

6.877 std::remove_const< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.877.1 Detailed Description

template<typename _Tp>
struct std::remove_const< _Tp >

remove_const
The documentation for this struct was generated from the following file:

- type_traits

6.878 std::remove_cv< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.878.1 Detailed Description

template<typename _Tp>
struct std::remove_cv< _Tp >

remove_cv
The documentation for this struct was generated from the following file:

- type_traits

6.879 std::remove_extent< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.879.1 Detailed Description

template<typename _Tp>
struct std::remove_extent< _Tp >

remove_extent
The documentation for this struct was generated from the following file:

- type_traits

6.880 std::remove_pointer< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.880.1 Detailed Description

template<**typename _Tp**>
struct std::remove_pointer< **_Tp** >

remove_pointer
The documentation for this struct was generated from the following file:

- type_traits

6.881 std::remove_reference< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.881.1 Detailed Description

template<**typename _Tp**>
struct std::remove_reference< **_Tp** >

remove_reference
The documentation for this struct was generated from the following file:

- type_traits

6.882 std::remove_volatile< _Tp > Struct Template Reference

`#include <type_traits>`

Public Types

- using **type**

6.882.1 Detailed Description

template<**typename _Tp**>
struct std::remove_volatile< **_Tp** >

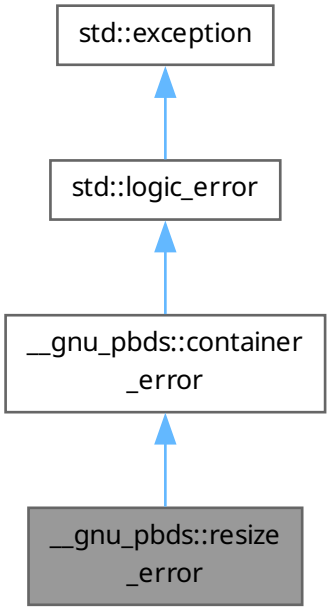
remove_volatile
The documentation for this struct was generated from the following file:

- type_traits

6.883 __gnu_pbds::resize_error Struct Reference

`#include <exception.hpp>`

Inheritance diagram for __gnu_pbds::resize_error:



Public Member Functions

- virtual const char ∗ what () const noexcept

6.883.1 Detailed Description

A container cannot be resized.

6.883.2 Member Function Documentation

what()

```
virtual const char * std::logic_error::what () const  [virtual], [noexcept], [inherited]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::future_error.
References what().
Referenced by what().
The documentation for this struct was generated from the following file:

- exception.hpp

6.884 __gnu_pbds::detail::resize_policy< _Tp > Class Template Reference

```
#include <resize_policy.hpp>
```

Public Types

- typedef _Tp **size_type**

Public Member Functions

- **resize_policy** (const resize_policy &other)
- size_type **get_new_size_for_arbitrary** (size_type) const
- size_type **get_new_size_for_grow** () const
- size_type **get_new_size_for_shrink** () const
- bool **grow_needed** (size_type) const
- void **notify_arbitrary** (size_type)
- void **notify_grow_resize** ()
- void **notify_shrink_resize** ()
- bool **resize_needed_for_grow** (size_type) const
- bool **resize_needed_for_shrink** (size_type) const
- bool **shrink_needed** (size_type) const
- void **swap** (resize_policy< _Tp > &)

Static Public Attributes

- static const _Tp **min_size**

6.884.1 Detailed Description

template<typename _Tp>
class __gnu_pbds::detail::resize_policy< _Tp >

Resize policy for binary heap.
The documentation for this class was generated from the following file:

- resize_policy.hpp

6.885 std::result_of< _Signature > Struct Template Reference

6.885.1 Detailed Description

template<typename _Signature>
struct std::result_of< _Signature >

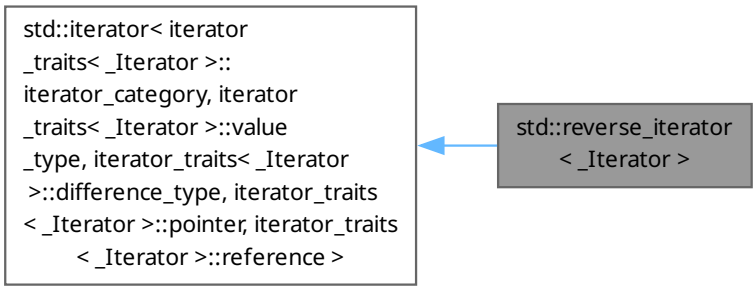
result_of
The documentation for this struct was generated from the following file:

- type_traits

6.886 std::reverse_iterator< _Iterator > Class Template Reference

`#include <stl_iterator.h>`

Inheritance diagram for std::reverse_iterator< _Iterator >:



Public Types

- using **difference_type**
- using **iterator_category**
- using **iterator_concept**
- typedef _Iterator **iterator_type**
- typedef __traits_type::pointer **pointer**
- using **reference**
- using **value_type**

Public Member Functions

- constexpr reverse_iterator () noexcept(/*conditional */)
- constexpr reverse_iterator (const reverse_iterator &__x) noexcept(/*conditional */)
- template<typename _Iter>
  constexpr reverse_iterator (const reverse_iterator< _Iter > &__x) noexcept(/*conditional */)
- constexpr reverse_iterator (iterator_type __x) noexcept(/*conditional */)
- constexpr iterator_type base () const noexcept(/*conditional */)
- constexpr reference operator* () const
- constexpr reverse_iterator operator+ (difference_type __n) const
- constexpr reverse_iterator & operator++ ()
- constexpr reverse_iterator operator++ (int)
- constexpr reverse_iterator & operator+= (difference_type __n)
- constexpr reverse_iterator operator- (difference_type __n) const
- constexpr reverse_iterator & operator-- ()
- constexpr reverse_iterator operator-- (int)
- constexpr reverse_iterator & operator-= (difference_type __n)
- constexpr pointer operator-> () const
- reverse_iterator & **operator=** (const reverse_iterator &)=default
- template<typename _Iter>
  constexpr reverse_iterator & **operator=** (const reverse_iterator< _Iter > &__x) noexcept(/*conditional */)
- constexpr reference operator[ ] (difference_type __n) const

Protected Types

- typedef iterator_traits< _Iterator > **__traits_type**

Protected Attributes

- _Iterator **current**

Friends

- constexpr iter_rvalue_reference_t< _Iterator > **iter_move** (const reverse_iterator &__i) noexcept(is_nothrow_↩ copy_constructible_v< _Iterator > &&noexcept(ranges::iter_move(--std::declval< _Iterator & >())))
- template<indirectly_swappable< _Iterator > _Iter2>
  constexpr void **iter_swap** (const reverse_iterator &__x, const reverse_iterator< _Iter2 > &__y) noexcept(is_↩ nothrow_copy_constructible_v< _Iterator > &&is_nothrow_copy_constructible_v< _Iter2 > &&noexcept(ranges::↩ iter_swap(--std::declval< _Iterator & >(), --std::declval< _Iter2 & >())))

6.886.1 Detailed Description

template<typename _Iterator>
class std::reverse_iterator< _Iterator >

Bidirectional and random access iterators have corresponding reverse iterator adaptors that iterate through the data structure in the opposite direction. They have the same signatures as the corresponding iterators. The fundamental relation between a reverse iterator and its corresponding iterator `i` is established by the identity:

```
&*(reverse_iterator(i)) == &*(i - 1)
```

This mapping is dictated by the fact that while there is always a pointer past the end of an array, there might not be a valid pointer before the beginning of an array. [24.4.1]/1,2

Reverse iterators can be tricky and surprising at first. Their semantics make sense, however, and the trickiness is a side effect of the requirement that the iterators must be safe.

6.886.2 Constructor & Destructor Documentation

reverse_iterator() [1/4]

```
template<typename _Iterator>
std::reverse_iterator< _Iterator >::reverse_iterator ()  [inline], [constexpr], [noexcept]
```

The default constructor value-initializes member `current`. If it is a pointer, that means it is zero-initialized.

Referenced by std::reverse_iterator< iterator >::operator+(), and std::reverse_iterator< iterator >::operator-().

reverse_iterator() [2/4]

```
template<typename _Iterator>
std::reverse_iterator< _Iterator >::reverse_iterator (
            iterator_type __x)  [inline], [explicit], [constexpr], [noexcept]
```

This iterator will move in the opposite direction that `x` does.

reverse_iterator() [3/4]

```
template<typename _Iterator>
std::reverse_iterator< _Iterator >::reverse_iterator (
            const reverse_iterator< _Iterator > & __x)  [inline], [constexpr], [noexcept]
```

The copy constructor is normal.

reverse_iterator() [4/4]

```
template<typename _Iterator>
template<typename _Iter>
std::reverse_iterator< _Iterator >::reverse_iterator (
            const reverse_iterator< _Iter > & __x)  [inline], [constexpr], [noexcept]
```

A reverse_iterator across other types can be copied if the underlying iterator can be converted to the type of `current`.

6.886.3 Member Function Documentation

base()

```
template<typename _Iterator>
iterator_type std::reverse_iterator< _Iterator >::base () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns

`current`, the iterator used for underlying work.

Referenced by std::operator==().

operator*()

```
template<typename _Iterator>
reference std::reverse_iterator< _Iterator >::operator* () const  [inline], [nodiscard], [constexpr]
```

Returns

A reference to the value at `--current`

This requires that `--current` is dereferenceable.

Warning

This implementation requires that for an iterator of the underlying iterator type, `x`, a reference obtained by `*x` remains valid after `x` has been modified or destroyed. This is a bug: http://gcc.gnu.org/PR51823

operator+()

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator+ (
            difference_type __n) const  [inline], [nodiscard], [constexpr]
```

Returns

A reverse_iterator that refers to `current` - *__n*

The underlying iterator must be a Random Access Iterator.

operator++() [1/2]

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator++ ()  [inline], [constexpr]
```

Returns

`*this`

Decrements the underlying iterator.

operator++() [2/2]

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator++ (
            int )  [inline], [constexpr]
```

Returns

The original value of `*this`

Decrements the underlying iterator.

operator+=()

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator+= (
            difference_type __n)  [inline], [constexpr]
```

Returns

*this

Moves the underlying iterator backwards *__n* steps. The underlying iterator must be a Random Access Iterator.

operator-()

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator- (
            difference_type __n) const  [inline], [nodiscard], [constexpr]
```

Returns

A reverse_iterator that refers to `current` - *__n*

The underlying iterator must be a Random Access Iterator.

operator--() [1/2]

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator-- ()  [inline], [constexpr]
```

Returns

`*this`

Increments the underlying iterator.

operator--() [2/2]

```
template<typename _Iterator>
reverse_iterator std::reverse_iterator< _Iterator >::operator-- (
            int )  [inline], [constexpr]
```

Returns

A reverse_iterator with the previous value of `*this`

Increments the underlying iterator.

operator-=()

```
template<typename _Iterator>
reverse_iterator & std::reverse_iterator< _Iterator >::operator-= (
            difference_type __n)  [inline], [constexpr]
```

Returns

∗this

Moves the underlying iterator forwards __*n* steps. The underlying iterator must be a Random Access Iterator.

operator->()

```
template<typename _Iterator>
pointer std::reverse_iterator< _Iterator >::operator-> () const  [inline], [nodiscard], [constexpr]
```

Returns

A pointer to the value at `--current`

This requires that `--current` is dereferenceable.

operator[]()

```
template<typename _Iterator>
reference std::reverse_iterator< _Iterator >::operator[] (
            difference_type __n) const  [inline], [nodiscard], [constexpr]
```

Returns

The value at `current` - __*n* - 1

The underlying iterator must be a Random Access Iterator.
The documentation for this class was generated from the following file:

- bits/stl_iterator.h

6.887 __gnu_cxx::rope< _CharT, _Alloc > Class Template Reference

`#include <rope>`

Public Types

- typedef _Rope_RopeConcatenation< _CharT, _Alloc > **__C**
- typedef _Rope_RopeFunction< _CharT, _Alloc > **__F**
- typedef _Rope_RopeLeaf< _CharT, _Alloc > **__L**
- typedef _Rope_RopeSubstring< _CharT, _Alloc > **__S**
- typedef __alloc_traits< _Alloc >::template rebind< __C >::other **_CAlloc**
- typedef __alloc_traits< _Alloc >::template rebind< _CharT >::other **_DataAlloc**
- typedef __alloc_traits< _Alloc >::template rebind< __F >::other **_FAlloc**
- typedef __alloc_traits< _Alloc >::template rebind< __L >::other **_LAlloc**
- typedef __alloc_traits< _Alloc >::template rebind< __S >::other **_SAlloc**
- typedef _Rope_const_iterator< _CharT, _Alloc > **const_iterator**
- typedef const _CharT ∗ **const_pointer**
- typedef _CharT **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef std::ptrdiff_t **difference_type**

- typedef _Rope_iterator< _CharT, _Alloc > **iterator**
- typedef _Rope_char_ptr_proxy< _CharT, _Alloc > **pointer**
- typedef _Rope_char_ref_proxy< _CharT, _Alloc > **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef std::size_t **size_type**
- typedef _CharT **value_type**

Public Member Functions

- **rope** (_CharT __c, const allocator_type &__a=allocator_type())
- **rope** (char_producer< _CharT > *__fn, size_type __len, bool __delete_fn, const allocator_type &__a=allocator_type())
- **rope** (const _CharT *__s, const _CharT *__e, const allocator_type &__a=allocator_type())
- **rope** (const _CharT *__s, const allocator_type &__a=allocator_type())
- **rope** (const _CharT *__s, size_type __len, const allocator_type &__a=allocator_type())
- **rope** (const allocator_type &__a=allocator_type())
- **rope** (const const_iterator &__s, const const_iterator &__e, const allocator_type &__a=allocator_type())
- **rope** (const iterator &__s, const iterator &__e, const allocator_type &__a=allocator_type())
- **rope** (const rope &__x, const allocator_type &__a=allocator_type())
- **rope** (size_type __n, _CharT __c, const allocator_type &__a=allocator_type())
- allocator_type & **_M_get_allocator** ()
- const allocator_type & **_M_get_allocator** () const
- rope & **append** ()
- rope & **append** (_CharT __c)
- rope & **append** (const _CharT *__c_string)
- rope & **append** (const _CharT *__iter, size_type __n)
- rope & **append** (const _CharT *__s, const _CharT *__e)
- rope & **append** (const rope &__y)
- rope & **append** (const_iterator __s, const_iterator __e)
- rope & **append** (size_type __n, _CharT __c)
- void **apply_to_pieces** (size_type __begin, size_type __end, _Rope_char_consumer< _CharT > &__c) const
- _CharT **at** (size_type __pos) const
- _CharT **back** () const
- void **balance** ()
- const_iterator **begin** ()
- const_iterator **begin** () const
- const _CharT * **c_str** () const
- void **clear** ()
- int **compare** (const rope &__y) const
- const_iterator **const_begin** () const
- const_iterator **const_end** () const
- const_reverse_iterator **const_rbegin** () const
- const_reverse_iterator **const_rend** () const
- void **copy** (_CharT *__buffer) const
- size_type **copy** (size_type __pos, size_type __n, _CharT *__buffer) const
- void **delete_c_str** ()
- void **dump** ()
- bool **empty** () const
- const_iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (const iterator &__p)

- iterator **erase** (const iterator &__p, const iterator &__q)
- void **erase** (size_type __p, size_type __n)
- size_type **find** (_CharT __c, size_type __pos=0) const
- size_type **find** (const _CharT *__s, size_type __pos=0) const
- _CharT **front** () const
- allocator_type **get_allocator** () const
- iterator **insert** (const iterator &__p)
- iterator **insert** (const iterator &__p, _CharT __c)
- iterator **insert** (const iterator &__p, const _CharT *__i, const _CharT *__j)
- iterator **insert** (const iterator &__p, const _CharT *__i, size_type __n)
- iterator **insert** (const iterator &__p, const _CharT *c_string)
- iterator **insert** (const iterator &__p, const const_iterator &__i, const const_iterator &__j)
- iterator **insert** (const iterator &__p, const iterator &__i, const iterator &__j)
- iterator **insert** (const iterator &__p, const rope &__r)
- iterator **insert** (const iterator &__p, size_type __n, _CharT __c)
- void **insert** (size_type __p)
- void **insert** (size_type __p, _CharT __c)
- void **insert** (size_type __p, const _CharT *__c_string)
- void **insert** (size_type __p, const _CharT *__i, const _CharT *__j)
- void **insert** (size_type __p, const _CharT *__i, size_type __n)
- void **insert** (size_type __p, const const_iterator &__i, const const_iterator &__j)
- void **insert** (size_type __p, const iterator &__i, const iterator &__j)
- void **insert** (size_type __p, const rope &__r)
- void **insert** (size_type __p, size_type __n, _CharT __c)
- size_type **length** () const
- size_type **max_size** () const
- iterator **mutable_begin** ()
- iterator **mutable_end** ()
- reverse_iterator **mutable_rbegin** ()
- reference **mutable_reference_at** (size_type __pos)
- reverse_iterator **mutable_rend** ()
- rope & **operator=** (const rope &__x)
- _CharT **operator[ ]** (size_type __pos) const
- void **pop_back** ()
- void **pop_front** ()
- void **push_back** (_CharT __x)
- void **push_front** (_CharT __x)
- const_reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- const_reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- void **replace** (const iterator &__p, _CharT __c)
- void **replace** (const iterator &__p, const _CharT *__c_string)
- void **replace** (const iterator &__p, const _CharT *__i, const _CharT *__j)
- void **replace** (const iterator &__p, const _CharT *__i, size_type __n)
- void **replace** (const iterator &__p, const iterator &__q, _CharT __c)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__c_string)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__i, const _CharT *__j)
- void **replace** (const iterator &__p, const iterator &__q, const _CharT *__i, size_type __n)
- void **replace** (const iterator &__p, const iterator &__q, const const_iterator &__i, const const_iterator &__j)
- void **replace** (const iterator &__p, const iterator &__q, const iterator &__i, const iterator &__j)

- void **replace** (const iterator &__p, const iterator &__q, const rope &__r)
- void **replace** (const iterator &__p, const rope &__r)
- void **replace** (const iterator &__p, const_iterator __i, const_iterator __j)
- void **replace** (const iterator &__p, iterator __i, iterator __j)
- void **replace** (size_type __p, _CharT __c)
- void **replace** (size_type __p, const _CharT *__c_string)
- void **replace** (size_type __p, const _CharT *__i, const _CharT *__j)
- void **replace** (size_type __p, const _CharT *__i, size_type __i_len)
- void **replace** (size_type __p, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_type __p, const iterator &__i, const iterator &__j)
- void **replace** (size_type __p, const rope &__r)
- void **replace** (size_type __p, size_type __n, _CharT __c)
- void **replace** (size_type __p, size_type __n, const _CharT *__c_string)
- void **replace** (size_type __p, size_type __n, const _CharT *__i, const _CharT *__j)
- void **replace** (size_type __p, size_type __n, const _CharT *__i, size_type __i_len)
- void **replace** (size_type __p, size_type __n, const const_iterator &__i, const const_iterator &__j)
- void **replace** (size_type __p, size_type __n, const iterator &__i, const iterator &__j)
- void **replace** (size_type __p, size_type __n, const rope &__r)
- const _CharT * **replace_with_c_str** ()
- size_type **size** () const
- rope< _CharT, _Alloc > **substr** (const_iterator __start)
- rope **substr** (const_iterator __start, const_iterator __end) const
- rope **substr** (iterator __start) const
- rope **substr** (iterator __start, iterator __end) const
- rope **substr** (size_type __start, size_type __len=1) const
- void **swap** (rope &__b)

Static Public Member Functions

- static __C * **_C_allocate** (std::size_t __n)
- static void **_C_deallocate** (__C *__p, std::size_t __n)
- static _CharT * **_Data_allocate** (std::size_t __n)
- static void **_Data_deallocate** (_CharT *__p, std::size_t __n)
- static __F * **_F_allocate** (std::size_t __n)
- static void **_F_deallocate** (__F *__p, std::size_t __n)
- static __L * **_L_allocate** (std::size_t __n)
- static void **_L_deallocate** (__L *__p, std::size_t __n)
- static __S * **_S_allocate** (std::size_t __n)
- static void **_S_deallocate** (__S *__p, std::size_t __n)

Public Attributes

- _RopeRep * **_M_tree_ptr**

Static Public Attributes

- static const size_type **npos**

Protected Types

- enum { **_S_copy_max** }
- typedef _Rope_base< _CharT, _Alloc > **_Base**
- typedef _CharT * **_Cstrptr**
- typedef _Rope_RopeConcatenation< _CharT, _Alloc > **_RopeConcatenation**
- typedef _Rope_RopeFunction< _CharT, _Alloc > **_RopeFunction**
- typedef _Rope_RopeLeaf< _CharT, _Alloc > **_RopeLeaf**
- typedef _Rope_RopeRep< _CharT, _Alloc > **_RopeRep**
- typedef _Rope_RopeSubstring< _CharT, _Alloc > **_RopeSubstring**
- typedef _Rope_self_destruct_ptr< _CharT, _Alloc > **_Self_destruct_ptr**
- typedef _Base::allocator_type **allocator_type**

Static Protected Member Functions

- static size_type **_S_allocated_capacity** (size_type __n)
- static bool **_S_apply_to_pieces** (_Rope_char_consumer< _CharT > &__c, const _RopeRep *__r, size_type __begin, size_type __end)
- static _RopeRep * **_S_concat** (_RopeRep *__left, _RopeRep *__right)
- static _RopeRep * **_S_concat_char_iter** (_RopeRep *__r, const _CharT *__iter, size_type __slen, allocator_type &__a)
- static _RopeRep * **_S_destr_concat_char_iter** (_RopeRep *__r, const _CharT *__iter, size_type __slen, allocator_type &__a)
- static _RopeLeaf * **_S_destr_leaf_concat_char_iter** (_RopeLeaf *__r, const _CharT *__iter, size_type __slen)
- static _CharT **_S_fetch** (_RopeRep *__r, size_type __pos)
- static _CharT * **_S_fetch_ptr** (_RopeRep *__r, size_type __pos)
- static bool **_S_is0** (_CharT __c)
- static _RopeLeaf * **_S_leaf_concat_char_iter** (_RopeLeaf *__r, const _CharT *__iter, size_type __slen)
- static _RopeConcatenation * **_S_new_RopeConcatenation** (_RopeRep *__left, _RopeRep *__right, allocator_type &__a)
- static _RopeFunction * **_S_new_RopeFunction** (char_producer< _CharT > *__f, size_type __size, bool __d, allocator_type &__a)
- static _RopeLeaf * **_S_new_RopeLeaf** (_CharT *__s, size_type __size, allocator_type &__a)
- static _RopeSubstring * **_S_new_RopeSubstring** (_Rope_RopeRep< _CharT, _Alloc > *__b, size_type __s, size_type __l, allocator_type &__a)
- static void **_S_ref** (_RopeRep *__t)
- static _RopeLeaf * **_S_RopeLeaf_from_unowned_char_ptr** (const _CharT *__s, size_type __size, allocator_type &__a)
- static size_type **_S_rounded_up_size** (size_type __n)
- static _RopeRep * **_S_substring** (_RopeRep *__base, size_type __start, size_type __endp1)
- static _RopeRep * **_S_tree_concat** (_RopeRep *__left, _RopeRep *__right)
- static void **_S_unref** (_RopeRep *__t)
- static _RopeRep * **replace** (_RopeRep *__old, size_type __pos1, size_type __pos2, _RopeRep *__r)

Static Protected Attributes

- static _CharT **_S_empty_c_str** [1]

Friends

- class **_Rope_char_ptr_proxy**< **_CharT, _Alloc** >
- class **_Rope_char_ref_proxy**< **_CharT, _Alloc** >
- class **_Rope_const_iterator**< **_CharT, _Alloc** >
- class **_Rope_iterator**< **_CharT, _Alloc** >
- class **_Rope_iterator_base**< **_CharT, _Alloc** >
- struct **_Rope_RopeRep**< **_CharT, _Alloc** >
- struct **_Rope_RopeSubstring**< **_CharT, _Alloc** >
- template<class _CharT2, class _Alloc2>
  rope< _CharT2, _Alloc2 > **operator+** (const rope< _CharT2, _Alloc2 > &__left, _CharT2 __right)
- template<class _CharT2, class _Alloc2>
  rope< _CharT2, _Alloc2 > **operator+** (const rope< _CharT2, _Alloc2 > &__left, const _CharT2 *__right)
- template<class _CharT2, class _Alloc2>
  rope< _CharT2, _Alloc2 > **operator+** (const rope< _CharT2, _Alloc2 > &__left, const rope< _CharT2, _Alloc2 > &__right)

6.887.1 Detailed Description

template<class _CharT, class _Alloc>
class __gnu_cxx::rope< **_CharT, _Alloc** >

This is an SGI extension.

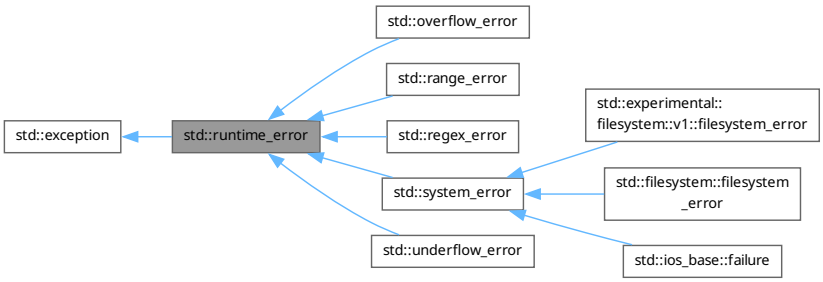
Todo Needs documentation! See http://gcc.gnu.org/onlinedocs/libstdc++/manual/documentation_style.html

The documentation for this class was generated from the following files:

- rope
- ropeimpl.h

6.888 std::runtime_error Class Reference

`#include <stdexcept_except.h>`
Inheritance diagram for std::runtime_error:



Public Member Functions

- **runtime_error** (const char *)
- **runtime_error** (const runtime_error &) noexcept

- runtime_error (const string &__arg)
- **runtime_error** (runtime_error &&) noexcept
- runtime_error & **operator=** (const runtime_error &) noexcept
- runtime_error & **operator=** (runtime_error &&) noexcept
- virtual const char * what () const noexcept

6.888.1 Detailed Description

One of two subclasses of exception.
Runtime errors represent problems outside the scope of a program; they cannot be easily predicted and can generally only be caught as the program executes.

6.888.2 Constructor & Destructor Documentation

runtime_error()

```
std::runtime_error::runtime_error (
            const string & __arg)  [explicit]
```

Takes a character string describing the error.
References runtime_error().
Referenced by runtime_error().

6.888.3 Member Function Documentation

what()

```
virtual const char * std::runtime_error::what () const  [virtual], [noexcept]
```

Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::experimental::filesystem::v1::filesystem_error, std::filesystem::filesystem_error, and std::ios_base::failure.
References what().
Referenced by what().
The documentation for this class was generated from the following file:

- stdexcept_except.h

6.889 __gnu_pbds::sample_probe_fn Class Reference

```
#include <sample_probe_fn.hpp>
```

Public Types

- typedef std::size_t **size_type**

Public Member Functions

- sample_probe_fn ()
- sample_probe_fn (const sample_probe_fn &)
- void swap (sample_probe_fn &)

Protected Member Functions

- size_type operator() (key_const_reference r_key, size_type i) const

6.889.1 Detailed Description

A sample probe policy.

6.889.2 Constructor & Destructor Documentation

sample_probe_fn() [1/2]

```
__gnu_pbds::sample_probe_fn::sample_probe_fn ()
```

Default constructor.
Referenced by sample_probe_fn(), and swap().

sample_probe_fn() [2/2]

```
__gnu_pbds::sample_probe_fn::sample_probe_fn (
            const sample_probe_fn & )
```

Copy constructor.
References sample_probe_fn().

6.889.3 Member Function Documentation

operator()()

```
size_type __gnu_pbds::sample_probe_fn::operator() (
            key_const_reference r_key,
            size_type i) const  [inline], [protected]
```

Returns the i-th offset from the hash value of some key r_key.

swap()

```
void __gnu_pbds::sample_probe_fn::swap (
            sample_probe_fn & )  [inline]
```

Swaps content.
References sample_probe_fn().
The documentation for this class was generated from the following file:

- sample_probe_fn.hpp

6.890 __gnu_pbds::sample_range_hashing Class Reference

```
#include <sample_range_hashing.hpp>
```

Public Types

- typedef std::size_t size_type

Public Member Functions

- sample_range_hashing ()
- sample_range_hashing (const sample_range_hashing &other)
- void swap (sample_range_hashing &other)

Protected Member Functions

- void notify_resized (size_type)
- size_type operator() (size_type) const

6.890.1 Detailed Description

A sample range-hashing functor.

6.890.2 Member Typedef Documentation

size_type

```
typedef std::size_t __gnu_pbds::sample_range_hashing::size_type
```
Size type.

6.890.3 Constructor & Destructor Documentation

sample_range_hashing() [1/2]

```
__gnu_pbds::sample_range_hashing::sample_range_hashing ()
```
Default constructor.
Referenced by sample_range_hashing(), and swap().

sample_range_hashing() [2/2]

```
__gnu_pbds::sample_range_hashing::sample_range_hashing (
            const sample_range_hashing & other)
```
Copy constructor.
References sample_range_hashing().

6.890.4 Member Function Documentation

notify_resized()

```
void __gnu_pbds::sample_range_hashing::notify_resized (
            size_type )  [protected]
```
Notifies the policy object that the container's size has changed to argument's size.

operator()()

```
size_type __gnu_pbds::sample_range_hashing::operator() (
            size_type ) const  [inline], [protected]
```
Transforms the __hash value hash into a ranged-hash value.

swap()

```
void __gnu_pbds::sample_range_hashing::swap (
            sample_range_hashing & other)  [inline]
```
Swaps content.
References sample_range_hashing().
The documentation for this class was generated from the following file:

- sample_range_hashing.hpp

6.891 __gnu_pbds::sample_ranged_hash_fn Class Reference

```
#include <sample_ranged_hash_fn.hpp>
```

Public Types

- typedef std::size_t **size_type**

Public Member Functions

- sample_ranged_hash_fn ()
- sample_ranged_hash_fn (const sample_ranged_hash_fn &)
- void swap (sample_ranged_hash_fn &)

Protected Member Functions

- void notify_resized (size_type)
- size_type operator() (key_const_reference) const

6.891.1 Detailed Description

A sample ranged-hash functor.

6.891.2 Constructor & Destructor Documentation

sample_ranged_hash_fn() [1/2]

```
__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn ()
```

Default constructor.
Referenced by sample_ranged_hash_fn(), and swap().

sample_ranged_hash_fn() [2/2]

```
__gnu_pbds::sample_ranged_hash_fn::sample_ranged_hash_fn (
            const sample_ranged_hash_fn & )
```

Copy constructor.
References sample_ranged_hash_fn().

6.891.3 Member Function Documentation

notify_resized()

```
void __gnu_pbds::sample_ranged_hash_fn::notify_resized (
            size_type )  [protected]
```

Notifies the policy object that the container's __size has changed to size.

operator()()

```
size_type __gnu_pbds::sample_ranged_hash_fn::operator() (
            key_const_reference ) const  [inline], [protected]
```

Transforms key_const_reference into a position within the table.

swap()

```
void __gnu_pbds::sample_ranged_hash_fn::swap (
            sample_ranged_hash_fn & )  [inline]
```

Swaps content.
References sample_ranged_hash_fn().
The documentation for this class was generated from the following file:

- sample_ranged_hash_fn.hpp

6.892 __gnu_pbds::sample_ranged_probe_fn Class Reference

```
#include <sample_ranged_probe_fn.hpp>
```

Public Types

- typedef std::size_t **size_type**

Public Member Functions

- **sample_ranged_probe_fn** (const sample_ranged_probe_fn &)
- void **swap** (sample_ranged_probe_fn &)

Protected Member Functions

- void **notify_resized** (size_type)
- size_type **operator()** (key_const_reference, std::size_t, size_type) const

6.892.1 Detailed Description

A sample ranged-probe functor.
The documentation for this class was generated from the following file:

- sample_ranged_probe_fn.hpp

6.893 __gnu_pbds::sample_resize_policy Class Reference

`#include <sample_resize_policy.hpp>`

Public Types

- typedef std::size_t size_type

Public Member Functions

- sample_resize_policy ()
- sample_range_hashing (const sample_resize_policy &other)
- void swap (sample_resize_policy &other)

Protected Member Functions

- size_type get_new_size (size_type size, size_type num_used_e) const
- bool is_resize_needed () const
- void notify_cleared ()
- void notify_erase_search_collision ()
- void notify_erase_search_end ()
- void notify_erase_search_start ()
- void notify_erased (size_type num_e)
- void notify_find_search_collision ()
- void notify_find_search_end ()
- void notify_find_search_start ()
- void notify_insert_search_collision ()
- void notify_insert_search_end ()
- void notify_insert_search_start ()
- void notify_inserted (size_type num_e)
- void notify_resized (size_type new_size)

6.893.1 Detailed Description

A sample resize policy.

6.893.2 Member Typedef Documentation

size_type

```
typedef std::size_t __gnu_pbds::sample_resize_policy::size_type
```
Size type.

6.893.3 Constructor & Destructor Documentation

sample_resize_policy()

```
__gnu_pbds::sample_resize_policy::sample_resize_policy ()
```
Default constructor.
Referenced by sample_range_hashing(), and swap().

6.893.4 Member Function Documentation

get_new_size()

```
size_type __gnu_pbds::sample_resize_policy::get_new_size (
            size_type size,
            size_type num_used_e) const  [protected]
```
Queries what the new size should be.

is_resize_needed()

```
bool __gnu_pbds::sample_resize_policy::is_resize_needed () const  [inline], [protected]
```
Queries whether a resize is needed.

notify_cleared()

```
void __gnu_pbds::sample_resize_policy::notify_cleared ()  [protected]
```
Notifies the table was cleared.

notify_erase_search_collision()

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_collision ()  [inline], [protected]
```
Notifies a search encountered a collision.

notify_erase_search_end()

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_end ()  [inline], [protected]
```
Notifies a search ended.

notify_erase_search_start()

```
void __gnu_pbds::sample_resize_policy::notify_erase_search_start ()  [inline], [protected]
```
Notifies a search started.

notify_erased()

```
void __gnu_pbds::sample_resize_policy::notify_erased (
            size_type num_e)  [inline], [protected]
```
Notifies an element was erased.

notify_find_search_collision()

```
void __gnu_pbds::sample_resize_policy::notify_find_search_collision ()  [inline], [protected]
```
Notifies a search encountered a collision.

notify_find_search_end()

```
void __gnu_pbds::sample_resize_policy::notify_find_search_end ()  [inline], [protected]
```
Notifies a search ended.

notify_find_search_start()

```
void __gnu_pbds::sample_resize_policy::notify_find_search_start ()  [inline], [protected]
```
Notifies a search started.

notify_insert_search_collision()

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_collision ()  [inline], [protected]
```
Notifies a search encountered a collision.

notify_insert_search_end()

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_end ()  [inline], [protected]
```
Notifies a search ended.

notify_insert_search_start()

```
void __gnu_pbds::sample_resize_policy::notify_insert_search_start ()  [inline], [protected]
```
Notifies a search started.

notify_inserted()

```
void __gnu_pbds::sample_resize_policy::notify_inserted (
            size_type num_e)  [inline], [protected]
```
Notifies an element was inserted.

notify_resized()

```
void __gnu_pbds::sample_resize_policy::notify_resized (
            size_type new_size)  [protected]
```
Notifies the table was resized to new_size.

sample_range_hashing()

```
__gnu_pbds::sample_resize_policy::sample_range_hashing (
            const sample_resize_policy & other)
```
Copy constructor.
References sample_resize_policy().

swap()

```
void __gnu_pbds::sample_resize_policy::swap (
            sample_resize_policy & other)  [inline]
```
Swaps content.
References sample_resize_policy().
The documentation for this class was generated from the following file:

- sample_resize_policy.hpp

6.894 __gnu_pbds::sample_resize_trigger Class Reference

`#include <sample_resize_trigger.hpp>`

Public Types

- typedef std::size_t size_type

Public Member Functions

- sample_resize_trigger ()
- sample_range_hashing (const sample_resize_trigger &)
- void swap (sample_resize_trigger &)

Protected Member Functions

- bool is_grow_needed (size_type size, size_type num_entries) const
- bool is_resize_needed () const
- void notify_cleared ()
- void notify_erase_search_collision ()
- void notify_erase_search_end ()
- void notify_erase_search_start ()
- void notify_erased (size_type num_entries)
- void notify_externally_resized (size_type new_size)
- void notify_find_search_collision ()
- void notify_find_search_end ()
- void notify_find_search_start ()
- void notify_insert_search_collision ()
- void notify_insert_search_end ()
- void notify_insert_search_start ()
- void notify_inserted (size_type num_entries)
- void notify_resized (size_type new_size)

6.894.1 Detailed Description

A sample resize trigger policy.

6.894.2 Member Typedef Documentation

size_type

`typedef std::size_t __gnu_pbds::sample_resize_trigger::size_type`
Size type.

6.894.3 Constructor & Destructor Documentation

sample_resize_trigger()

`__gnu_pbds::sample_resize_trigger::sample_resize_trigger ()`
Default constructor.
Referenced by sample_range_hashing(), and swap().

6.894.4 Member Function Documentation

is_grow_needed()

```
bool __gnu_pbds::sample_resize_trigger::is_grow_needed (
            size_type size,
            size_type num_entries) const  [inline], [protected]
```

Queries whether a grow is needed.

is_resize_needed()

```
bool __gnu_pbds::sample_resize_trigger::is_resize_needed () const  [inline], [protected]
```

Queries whether a resize is needed.

notify_cleared()

```
void __gnu_pbds::sample_resize_trigger::notify_cleared ()  [protected]
```

Notifies the table was cleared.

notify_erase_search_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_collision ()  [inline], [protected]
```

Notifies a search encountered a collision.

notify_erase_search_end()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_end ()  [inline], [protected]
```

Notifies a search ended.

notify_erase_search_start()

```
void __gnu_pbds::sample_resize_trigger::notify_erase_search_start ()  [inline], [protected]
```

Notifies a search started.

notify_erased()

```
void __gnu_pbds::sample_resize_trigger::notify_erased (
            size_type num_entries)  [inline], [protected]
```

Notifies an element was erased.

notify_externally_resized()

```
void __gnu_pbds::sample_resize_trigger::notify_externally_resized (
            size_type new_size)  [protected]
```

Notifies the table was resized externally.

notify_find_search_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_collision ()  [inline], [protected]
```

Notifies a search encountered a collision.

notify_find_search_end()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_end ()  [inline], [protected]
```

Notifies a search ended.

notify_find_search_start()

```
void __gnu_pbds::sample_resize_trigger::notify_find_search_start ()  [inline], [protected]
```
Notifies a search started.

notify_insert_search_collision()

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_collision ()  [inline], [protected]
```
Notifies a search encountered a collision.

notify_insert_search_end()

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_end ()  [inline], [protected]
```
Notifies a search ended.

notify_insert_search_start()

```
void __gnu_pbds::sample_resize_trigger::notify_insert_search_start ()  [inline], [protected]
```
Notifies a search started.

notify_inserted()

```
void __gnu_pbds::sample_resize_trigger::notify_inserted (
            size_type num_entries)  [inline], [protected]
```
Notifies an element was inserted. the total number of entries in the table is num_entries.

notify_resized()

```
void __gnu_pbds::sample_resize_trigger::notify_resized (
            size_type new_size)  [protected]
```
Notifies the table was resized as a result of this object's signifying that a resize is needed.

sample_range_hashing()

```
__gnu_pbds::sample_resize_trigger::sample_range_hashing (
            const sample_resize_trigger & )
```
Copy constructor.
References sample_resize_trigger().

swap()

```
void __gnu_pbds::sample_resize_trigger::swap (
            sample_resize_trigger & )  [inline]
```
Swaps content.
References sample_resize_trigger().
The documentation for this class was generated from the following file:

- sample_resize_trigger.hpp

6.895 __gnu_pbds::sample_size_policy Class Reference

```
#include <sample_size_policy.hpp>
```

Public Types

- typedef std::size_t size_type

Public Member Functions

- sample_size_policy ()
- sample_range_hashing (const sample_size_policy &)
- void swap (sample_size_policy &other)

Protected Member Functions

- size_type get_nearest_larger_size (size_type size) const
- size_type get_nearest_smaller_size (size_type size) const

6.895.1 Detailed Description

A sample size policy.

6.895.2 Member Typedef Documentation

size_type

```
typedef std::size_t __gnu_pbds::sample_size_policy::size_type
```

Size type.

6.895.3 Constructor & Destructor Documentation

sample_size_policy()

```
__gnu_pbds::sample_size_policy::sample_size_policy ()
```

Default constructor.
Referenced by sample_range_hashing(), and swap().

6.895.4 Member Function Documentation

get_nearest_larger_size()

```
size_type __gnu_pbds::sample_size_policy::get_nearest_larger_size (
            size_type size) const  [inline], [protected]
```

Given a __size size, returns a __size that is larger.

get_nearest_smaller_size()

```
size_type __gnu_pbds::sample_size_policy::get_nearest_smaller_size (
            size_type size) const  [inline], [protected]
```

Given a __size size, returns a __size that is smaller.

sample_range_hashing()

```
__gnu_pbds::sample_size_policy::sample_range_hashing (
            const sample_size_policy & )
```

Copy constructor.
References sample_size_policy().

swap()

```
void __gnu_pbds::sample_size_policy::swap (
            sample_size_policy & other)  [inline]
```

Swaps content.
References sample_size_policy().

The documentation for this class was generated from the following file:

- sample_size_policy.hpp

6.896 __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc > Class Template Reference

`#include <sample_tree_node_update.hpp>`

6.896.1 Detailed Description

template<typename Const_Node_Iter, typename Node_Iter, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >

A sample node updator.
The documentation for this class was generated from the following file:

- sample_tree_node_update.hpp

6.897 __gnu_pbds::sample_trie_access_traits Struct Reference

`#include <sample_trie_access_traits.hpp>`

Public Types

- enum { **max_size** }
- typedef std::string::const_iterator **const_iterator**
- typedef char e_type
- typedef rebind_traits< _Alloc, key_type >::const_reference **key_const_reference**
- typedef std::string **key_type**
- typedef std::size_t **size_type**

Static Public Member Functions

- static const_iterator begin (key_const_reference)
- static size_type e_pos (e_type)
- static const_iterator end (key_const_reference)

6.897.1 Detailed Description

A sample trie element access traits.

6.897.2 Member Typedef Documentation

e_type

```
typedef char __gnu_pbds::sample_trie_access_traits::e_type
```
Element type.

6.897.3 Member Function Documentation

begin()

```
const_iterator __gnu_pbds::sample_trie_access_traits::begin (
            key_const_reference )  [inline], [static]
```
Returns a const_iterator to the first element of r_key.

e_pos()

```
size_type __gnu_pbds::sample_trie_access_traits::e_pos (
            e_type )  [inline], [static]
```

Maps an element to a position.

end()

```
const_iterator __gnu_pbds::sample_trie_access_traits::end (
            key_const_reference )  [inline], [static]
```

Returns a const_iterator to the after-last element of r_key.
The documentation for this struct was generated from the following file:

- sample_trie_access_traits.hpp

6.898 __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference

```
#include <sample_trie_node_update.hpp>
```

Public Types

- typedef std::size_t **metadata_type**

Protected Member Functions

- sample_trie_node_update ()
- void operator() (node_iterator, node_const_iterator) const

6.898.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

A sample node updator.

6.898.2 Constructor & Destructor Documentation

sample_trie_node_update()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
__gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::sample_trie_node_↩
update ()  [protected]
```

Default constructor.

6.898.3 Member Function Documentation

operator()()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
void __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::operator() (
            node_iterator ,
            node_const_iterator ) const  [inline], [protected]
```

Updates the rank of a node through a node_iterator node_it; end_nd_it is the end node iterator.
The documentation for this class was generated from the following file:

- sample_trie_node_update.hpp

6.899 __gnu_pbds::sample_update_policy Struct Reference

`#include <sample_update_policy.hpp>`

Public Member Functions

- sample_update_policy ()
- sample_update_policy (const sample_update_policy &)
- void swap (sample_update_policy &other)

Protected Types

- typedef some_metadata_type metadata_type

Protected Member Functions

- metadata_type operator() () const
- bool operator() (metadata_reference) const

6.899.1 Detailed Description

A sample list-update policy.

6.899.2 Member Typedef Documentation

metadata_type

```
typedef some_metadata_type __gnu_pbds::sample_update_policy::metadata_type  [protected]
```

Metadata on which this functor operates.

6.899.3 Constructor & Destructor Documentation

sample_update_policy() [1/2]

```
__gnu_pbds::sample_update_policy::sample_update_policy ()
```

Default constructor.
Referenced by sample_update_policy(), and swap().

sample_update_policy() [2/2]

```
__gnu_pbds::sample_update_policy::sample_update_policy (
            const sample_update_policy & )
```

Copy constructor.
References sample_update_policy().

6.899.4 Member Function Documentation

operator()() [1/2]

```
metadata_type __gnu_pbds::sample_update_policy::operator() () const  [protected]
```

Creates a metadata object.

operator()() [2/2]

```
bool __gnu_pbds::sample_update_policy::operator() (
            metadata_reference ) const  [protected]
```

Decides whether a metadata object should be moved to the front of the list. A list-update based containers object will call this method to decide whether to move a node to the front of the list. The method should return true if the node should be moved to the front of the list.

swap()

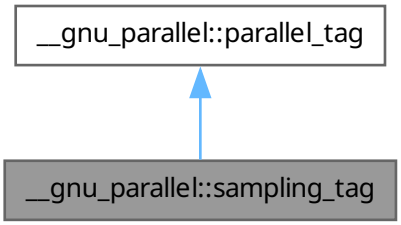
```
void __gnu_pbds::sample_update_policy::swap (
            sample_update_policy & other)  [inline]
```

Swaps content.
References sample_update_policy().
The documentation for this struct was generated from the following file:

- sample_update_policy.hpp

6.900 __gnu_parallel::sampling_tag Struct Reference

`#include <tags.h>`
Inheritance diagram for __gnu_parallel::sampling_tag:



Public Member Functions

- **sampling_tag** (_ThreadIndex __num_threads)
- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

6.900.1 Detailed Description

Forces parallel merging with exact splitting, at compile time.

6.900.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

6.901 std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs > Class Template Reference

`#include <scoped_allocator>`

Public Types

- typedef __traits::const_pointer **const_pointer**
- typedef __traits::const_void_pointer **const_void_pointer**
- typedef __traits::difference_type **difference_type**
- typedef __inner_type::__type **inner_allocator_type**
- typedef __and_< typename __traits::is_always_equal, typenameallocator_traits< _InnerAllocs >::is_always_equal... >::type **is_always_equal**
- typedef _OuterAlloc **outer_allocator_type**
- typedef __traits::pointer **pointer**
- typedef __or_< typename __traits::propagate_on_container_copy_assignment, typenameallocator_traits< _InnerAllocs >::propagate_on_container_copy_assignment... >::type **propagate_on_container_copy_assignment**
- typedef __or_< typename __traits::propagate_on_container_move_assignment, typenameallocator_traits< _InnerAllocs >::propagate_on_container_move_assignment... >::type **propagate_on_container_move_assignment**
- typedef __or_< typename __traits::propagate_on_container_swap, typenameallocator_traits< _InnerAllocs >::propagate_on_container_swap... >::type **propagate_on_container_swap**
- typedef __traits::size_type **size_type**
- typedef __traits::value_type **value_type**
- typedef __traits::void_pointer **void_pointer**

Public Member Functions

- template<typename _Outer2, typename = _Constructible<_Outer2>>
  scoped_allocator_adaptor (_Outer2 &&__outer, const _InnerAllocs &... __inner) noexcept
- **scoped_allocator_adaptor** (const scoped_allocator_adaptor &__other) noexcept
- template<typename _Outer2, typename = _Constructible<const _Outer2&>>
  scoped_allocator_adaptor (const scoped_allocator_adaptor< _Outer2, _InnerAllocs... > &__other) noexcept

- **scoped_allocator_adaptor** (scoped_allocator_adaptor &&__other) noexcept
- template<typename _Outer2, typename = _Constructible<_Outer2>>
  scoped_allocator_adaptor (scoped_allocator_adaptor< _Outer2, _InnerAllocs... > &&__other) noexcept
- pointer **allocate** (size_type __n)
- pointer **allocate** (size_type __n, const_void_pointer __hint)
- template<typename _Tp, typename... _Args>
  void **construct** (_Tp *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n) noexcept
- template<typename _Tp>
  void **destroy** (_Tp *__p)
- const inner_allocator_type & **inner_allocator** () const noexcept
- inner_allocator_type & **inner_allocator** () noexcept
- size_type **max_size** () const
- scoped_allocator_adaptor & **operator=** (const scoped_allocator_adaptor &)=default
- scoped_allocator_adaptor & **operator=** (scoped_allocator_adaptor &&)=default
- const outer_allocator_type & **outer_allocator** () const noexcept
- outer_allocator_type & **outer_allocator** () noexcept
- scoped_allocator_adaptor **select_on_container_copy_construction** () const

Friends

- template<typename...>
  struct **__inner_type_impl**
- template<typename _OutA1, typename _OutA2, typename... _InA>
  bool **operator==** (const scoped_allocator_adaptor< _OutA1, _InA... > &__a, const scoped_allocator_adaptor< _OutA2, _InA... > &__b) noexcept

Related Symbols

(Note that these are not member symbols.)

- template<typename _OutA1, typename _OutA2, typename... _InA>
  bool **operator==** (const scoped_allocator_adaptor< _OutA1, _InA... > &__a, const scoped_allocator_adaptor< _OutA2, _InA... > &__b) noexcept

6.901.1 Detailed Description

template<typename _OuterAlloc, typename... _InnerAllocs>
class std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >

An adaptor to recursively pass an allocator to the objects it constructs.
The documentation for this class was generated from the following file:

- scoped_allocator

6.902 std::seed_seq Class Reference

`#include <random>`

Public Types

- typedef uint_least32_t result_type

Public Member Functions

- seed_seq () noexcept
- template<typename _InputIterator>
  seed_seq (_InputIterator __begin, _InputIterator __end)
- **seed_seq** (const seed_seq &)=delete
- template<typename _IntType, typename = _Require<is_integral<_IntType>>>
  seed_seq (std::initializer_list< _IntType > __il)
- template<typename _RandomAccessIterator>
  void **generate** (_RandomAccessIterator __begin, _RandomAccessIterator __end)
- seed_seq & **operator=** (const seed_seq &)=delete
- template<typename _OutputIterator>
  void **param** (_OutputIterator __dest) const
- size_t **size** () const noexcept

6.902.1 Detailed Description

The seed_seq class generates sequences of seeds for random number generators.

Since

C++11

6.902.2 Member Typedef Documentation

result_type

```
typedef uint_least32_t std::seed_seq::result_type
```

The type of the seed vales.

6.902.3 Constructor & Destructor Documentation

seed_seq()

```
std::seed_seq::seed_seq ()  [inline], [noexcept]
```

Default constructor.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.903 __gnu_cxx::select1st< _Pair > Struct Template Reference

```
#include <functional>
```

Public Types

- typedef _Pair argument_type
- typedef _Pair::first_type result_type

Public Member Functions

- _Pair::first_type & **operator()** (_Pair &__x) const
- template<typename _Pair2>
  _Pair2::first_type & **operator()** (_Pair2 &__x) const
- const _Pair::first_type & **operator()** (const _Pair &__x) const
- template<typename _Pair2>
  const _Pair2::first_type & **operator()** (const _Pair2 &__x) const

6.903.1 Detailed Description

template<class _Pair>
struct __gnu_cxx::select1st< _Pair >

An SGI extension .

6.903.2 Member Typedef Documentation

argument_type

```
typedef _Pair std::unary_function< _Pair, _Pair::first_type >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _Pair::first_type std::unary_function< _Pair, _Pair::first_type >::result_type  [inherited]
```

`result_type` is the return type
The documentation for this struct was generated from the following file:

- ext/functional

6.904 __gnu_cxx::select2nd< _Pair > Struct Template Reference

```
#include <functional>
```

Public Types

- typedef _Pair argument_type
- typedef _Pair::second_type result_type

Public Member Functions

- _Pair::second_type & **operator()** (_Pair &__x) const
- const _Pair::second_type & **operator()** (const _Pair &__x) const

6.904.1 Detailed Description

template<class _Pair>
struct __gnu_cxx::select2nd< _Pair >

An SGI extension .

6.904.2 Member Typedef Documentation

argument_type

```
typedef _Pair std::unary_function< _Pair, _Pair::second_type >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _Pair::second_type std::unary_function< _Pair, _Pair::second_type >::result_type  [inherited]
```

`result_type` is the return type
The documentation for this struct was generated from the following file:

- ext/functional

6.905 __gnu_pbds::detail::select_value_type< Key, Mapped > Struct Template Reference

`#include <types_traits.hpp>`

Public Types

- typedef std::pair< const Key, Mapped > **type**

6.905.1 Detailed Description

template<typename Key, typename Mapped>
struct __gnu_pbds::detail::select_value_type< Key, Mapped >

Choose value_type to be a key/value pair or just a key.
The documentation for this struct was generated from the following file:

- types_traits.hpp

6.906 __gnu_pbds::detail::select_value_type< Key, null_type > Struct Template Reference

`#include <types_traits.hpp>`

Public Types

- typedef Key **type**

6.906.1 Detailed Description

template<typename Key>
struct __gnu_pbds::detail::select_value_type< Key, null_type >

Specialization for sets where the key is the value_type.
The documentation for this struct was generated from the following file:

- types_traits.hpp

6.907 std::basic_istream< _CharT, _Traits >::sentry Class Reference

`#include <istream>`

Public Types

- typedef __istream_type::__ctype_type **__ctype_type**
- typedef _Traits::int_type **__int_type**
- typedef basic_istream< _CharT, _Traits > **__istream_type**
- typedef basic_streambuf< _CharT, _Traits > **__streambuf_type**
- typedef _Traits traits_type

Public Member Functions

- sentry (basic_istream< _CharT, _Traits > &__is, bool __noskipws=false)
- operator bool () const

6.907.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_istream< _CharT, _Traits >::sentry

Performs setup work for input streams.
Objects of this class are created before all of the standard extractors are run. It is responsible for *exception-safe prefix and suffix operations,* although only prefix actions are currently required by the standard.

6.907.2 Member Typedef Documentation

traits_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_istream< _CharT, _Traits >::sentry::traits_type
```

Easy access to dependent types.

6.907.3 Constructor & Destructor Documentation

sentry()

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits >::sentry::sentry (
            basic_istream< _CharT, _Traits > & __is,
            bool __noskipws = false)  [explicit]
```

The constructor performs all the work.

Parameters

__is	The input stream to guard.
__noskipws	Whether to consume whitespace or not.

If the stream state is good (*__is.good()* is true), then the following actions are performed, otherwise the sentry state is false (*not okay*) and failbit is set in the stream state.
The sentry's preparatory actions are:

1. if the stream is tied to an output stream, `is.tie()->flush()` is called to synchronize the output sequence
2. if *__noskipws* is false, and `ios_base::skipws` is set in `is.flags()`, the sentry extracts and discards whitespace characters from the stream. The currently imbued locale is used to determine whether each character is whitespace.

If the stream state is still good, then the sentry state becomes true (*okay*).
References std::basic_istream< _CharT, _Traits >::basic_istream(), std::ios_base::badbit, std::basic_ios< _CharT, _Traits >::good(), and std::ios_base::goodbit.

6.907.4 Member Function Documentation

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits >::sentry::operator bool () const  [inline], [explicit]
```

Quick status checking.

Returns

The sentry state.

For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (true == okay).
The documentation for this class was generated from the following files:

- istream
- istream.tcc

6.908 std::basic_ostream< _CharT, _Traits >::sentry Class Reference

`#include <ostream.h>`

Public Member Functions

- sentry (basic_ostream< _CharT, _Traits > &__os)
- ~sentry ()
- operator bool () const

6.908.1 Detailed Description

template<typename _CharT, typename _Traits>
class std::basic_ostream< _CharT, _Traits >::sentry

Performs setup work for output streams.
Objects of this class are created before all of the standard inserters are run. It is responsible for *exception-safe prefix and suffix operations*.

6.908.2 Constructor & Destructor Documentation

sentry()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::sentry (
            basic_ostream< _CharT, _Traits > & __os)  [explicit]
```

The constructor performs preparatory work.

Parameters

__os	The output stream to guard.

If the stream state is good (*__os.good()* is true), then if the stream is tied to another output stream, `is.tie()->flush()` is called to synchronize the output sequences.
If the stream state is still good, then the sentry state becomes true (*okay*).
References std::basic_ostream< _CharT, _Traits >::basic_ostream(), std::basic_ios< _CharT, _Traits >::bad(), std::ios_base::failbit, std::basic_ios< _CharT, _Traits >::good(), std::basic_ios< _CharT, _Traits >::setstate(), and std::basic_ios< _CharT, _Traits >::tie().

~sentry()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::~sentry ()  [inline]
```

Possibly flushes the stream.
If `ios_base::unitbuf` is set in `os.flags()`, and `std::uncaught_exception()` is true, the sentry destructor flushes the output stream.
References std::ios_base::badbit, std::uncaught_exception(), and std::ios_base::unitbuf.

6.908.3 Member Function Documentation

operator bool()

```
template<typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits >::sentry::operator bool () const  [inline], [explicit]
```

Quick status checking.

Returns

The sentry state.

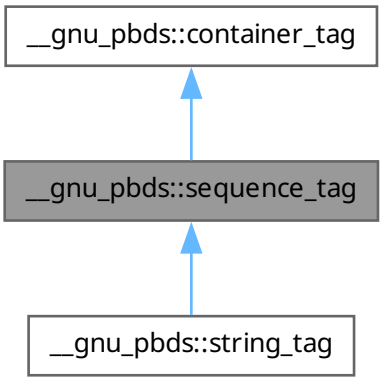
For ease of use, sentries may be converted to booleans. The return value is that of the sentry state (true == okay).
The documentation for this class was generated from the following files:

- ostream.h
- ostream.tcc

6.909 __gnu_pbds::sequence_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::sequence_tag:



6.909.1 Detailed Description

Basic sequence.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.910 __gnu_parallel::sequential_tag Struct Reference

```
#include <tags.h>
```

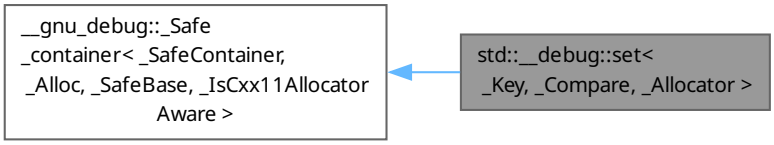
6.910.1 Detailed Description

Forces sequential execution at compile time.
The documentation for this struct was generated from the following file:

- tags.h

6.911 std::__debug::set< _Key, _Compare, _Allocator > Class Template Reference

`#include <set.h>`
Inheritance diagram for std::__debug::set< _Key, _Compare, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, set > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, set > **iterator**
- typedef _Compare **key_compare**
- typedef _Key **key_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Compare **value_compare**
- typedef _Key **value_type**

Public Member Functions

- **set** (_Base_ref __x)
- template<typename _InputIterator>
  set (_InputIterator __first, _InputIterator __last, const _Compare &__comp=_Compare(), const _Allocator &__a=_Allocator())
- template<typename _InputIterator>
  set (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- **set** (const _Compare &__comp, const _Allocator &__a=_Allocator())
- **set** (const allocator_type &__a)
- **set** (const set &)=default
- **set** (const set &__x, const __type_identity_t< allocator_type > &__a)

- **set** (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- **set** (initializer_list< value_type > __l, const allocator_type &__a)
- **set** (set &&)=default
- **set** (set &&__x, const __type_identity_t< allocator_type > &__a) noexcept(noexcept(_Base(std::move(__x), _a)))
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- void **clear** () noexcept
- const_reverse_iterator **crbegin** () const noexcept
- const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __pos, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- std::pair< iterator, iterator > **equal_range** (const key_type &__x)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__x) const
- _Base_iterator **erase** (_Base_const_iterator __position)
- template<__heterogeneous_tree_key< set > _Kt>
  size_type **erase** (_Kt &&__x)
- size_type **erase** (const key_type &__x)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __position)
- iterator **find** (const key_type &__x)
- const_iterator **find** (const key_type &__x) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- template<__heterogeneous_tree_key< set > _Kt>
  std::pair< iterator, bool > **insert** (_Kt &&__x)
- std::pair< iterator, bool > **insert** (const value_type &__x)
- template<__heterogeneous_tree_key< set > _Kt>
  iterator **insert** (const_iterator __position, _Kt &&__x)
- iterator **insert** (const_iterator __position, const value_type &__x)
- iterator **insert** (const_iterator __position, value_type &&__x)
- void **insert** (initializer_list< value_type > __l)
- std::pair< iterator, bool > **insert** (value_type &&__x)
- iterator **lower_bound** (const key_type &__x)
- const_iterator **lower_bound** (const key_type &__x) const
- set & **operator=** (const set &)=default
- set & **operator=** (initializer_list< value_type > __l)
- set & **operator=** (set &&)=default
- const_reverse_iterator **rbegin** () const noexcept
- reverse_iterator **rbegin** () noexcept
- const_reverse_iterator **rend** () const noexcept
- reverse_iterator **rend** () noexcept
- void **swap** (set &__x) noexcept(/*conditional */)
- iterator **upper_bound** (const key_type &__x)
- const_iterator **upper_bound** (const key_type &__x) const

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

6.911.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Allocator = std::allocator<_Key>>
class std::__debug::set< _Key, _Compare, _Allocator >

Class std::set with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- set.h

6.912 std::set< _Key, _Compare, _Alloc > Class Template Reference

`#include <set>`

Public Types

- typedef _Key key_type
- typedef _Key value_type
- typedef _Compare key_compare
- typedef _Compare value_compare
- typedef _Alloc allocator_type

- typedef _Alloc_traits::pointer pointer
- typedef _Alloc_traits::const_pointer const_pointer
- typedef _Alloc_traits::reference reference
- typedef _Alloc_traits::const_reference const_reference
- typedef _Rep_type::const_iterator iterator
- typedef _Rep_type::const_iterator const_iterator
- typedef _Rep_type::const_reverse_iterator reverse_iterator
- typedef _Rep_type::const_reverse_iterator const_reverse_iterator
- typedef _Rep_type::size_type size_type
- typedef _Rep_type::difference_type difference_type

Public Member Functions

- set ()=default
- template<typename _InputIterator>
  set (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  set (_InputIterator __first, _InputIterator __last, const _Compare &__comp, const allocator_type &__a=allocator_type())
- template<typename _InputIterator>
  set (_InputIterator __first, _InputIterator __last, const allocator_type &__a)

- set (const _Compare &__comp, const allocator_type &__a=allocator_type())
- set (const allocator_type &__a)
- set (const set &)=default
- set (const set &__x, const __type_identity_t< allocator_type > &__a)
- set (initializer_list< value_type > __l, const _Compare &__comp=_Compare(), const allocator_type &__a=allocator_type())
- set (initializer_list< value_type > __l, const allocator_type &__a)
- set (set &&)=default
- set (set &&__x, const __type_identity_t< allocator_type > &__a) noexcept(is_nothrow_copy_constructible< _Compare >::value &&_Alloc_traits::_S_always_equal())
- ~set ()=default
- iterator begin () const noexcept
- iterator cbegin () const noexcept
- iterator cend () const noexcept
- void clear () noexcept
- reverse_iterator crbegin () const noexcept
- reverse_iterator crend () const noexcept
- template<typename... _Args>
  std::pair< iterator, bool > emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- iterator end () const noexcept
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- iterator erase (const_iterator __position)
- allocator_type get_allocator () const noexcept
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- std::pair< iterator, bool > insert (const value_type &__x)
- void insert (initializer_list< value_type > __l)
- std::pair< iterator, bool > **insert** (value_type &&__x)
- key_compare key_comp () const
- size_type max_size () const noexcept
- set & operator= (const set &)=default
- set & operator= (initializer_list< value_type > __l)
- set & operator= (set &&)=default
- reverse_iterator rbegin () const noexcept
- reverse_iterator rend () const noexcept
- size_type size () const noexcept
- void swap (set &__x) noexcept(/*conditional */)
- value_compare value_comp () const

- iterator insert (const_iterator __position, const value_type &__x)
- iterator insert (const_iterator __position, value_type &&__x)

- size_type count (const key_type &__x) const

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_t._M_find_tr(__x), void(), true)

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const

- iterator lower_bound (const key_type &__x)
- const_iterator lower_bound (const key_type &__x) const

- iterator upper_bound (const key_type &__x)
- const_iterator upper_bound (const key_type &__x) const

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const

Friends

- template<typename _K1, typename _C1, typename _A1>
  bool **operator**< (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)
- template<typename _K1, typename _C1, typename _A1>
  bool **operator==** (const set< _K1, _C1, _A1 > &, const set< _K1, _C1, _A1 > &)

6.912.1 Detailed Description

template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
class std::set< _Key, _Compare, _Alloc >

A standard container made up of unique keys, which can be retrieved in logarithmic time.

Since

C++98

Template Parameters

_Key	Type of key objects.
_Compare	Comparison function object type, defaults to less<_Key>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a container, a reversible container, and an associative container (using unique keys).
Sets support bidirectional iterators.
The private tree data is declared exactly the same way for set and multiset; the distinction is made entirely in how the tree functions are called (*_unique versus *_equal, same as the standard).

6.912.2 Member Typedef Documentation

allocator_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Alloc std::set< _Key, _Compare, _Alloc >::allocator_type
```

Public typedefs.

const_iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Rep_type::const_iterator std::set< _Key, _Compare, _Alloc >::const_iterator
```

Iterator-related typedefs.

const_pointer

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Alloc_traits::const_pointer std::set< _Key, _Compare, _Alloc >::const_pointer
```

Iterator-related typedefs.

const_reference

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Alloc_traits::const_reference std::set< _Key, _Compare, _Alloc >::const_reference
```

Iterator-related typedefs.

const_reverse_iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::const_reverse_↩
iterator
```

Iterator-related typedefs.

difference_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Rep_type::difference_type std::set< _Key, _Compare, _Alloc >::difference_type
```

Iterator-related typedefs.

iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Rep_type::const_iterator std::set< _Key, _Compare, _Alloc >::iterator
```

Iterator-related typedefs.

key_compare

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
```

```
typedef _Compare std::set< _Key, _Compare, _Alloc >::key_compare
```
Public typedefs.

key_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Key std::set< _Key, _Compare, _Alloc >::key_type
```
Public typedefs.

pointer

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Alloc_traits::pointer std::set< _Key, _Compare, _Alloc >::pointer
```
Iterator-related typedefs.

reference

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Alloc_traits::reference std::set< _Key, _Compare, _Alloc >::reference
```
Iterator-related typedefs.

reverse_iterator

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Rep_type::const_reverse_iterator std::set< _Key, _Compare, _Alloc >::reverse_iterator
```
Iterator-related typedefs.

size_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Rep_type::size_type std::set< _Key, _Compare, _Alloc >::size_type
```
Iterator-related typedefs.

value_compare

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Compare std::set< _Key, _Compare, _Alloc >::value_compare
```
Public typedefs.

value_type

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
typedef _Key std::set< _Key, _Compare, _Alloc >::value_type
```
Public typedefs.

6.912.3 Constructor & Destructor Documentation

set() [1/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set ()  [default]
```

Default constructor creates no elements.

set() [2/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Creates a set with no elements.

Parameters

__comp	Comparator to use.
__a	An allocator object.

set() [3/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
std::set< _Key, _Compare, _Alloc >::set (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

Builds a set from a range.

Parameters

__first	An input iterator.
__last	An input iterator.

Create a set consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

set() [4/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
std::set< _Key, _Compare, _Alloc >::set (
            _InputIterator __first,
            _InputIterator __last,
            const _Compare & __comp,
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a set from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__comp	A comparison functor.
__a	An allocator object.

Create a set consisting of copies of the elements from [__first,__last). This is linear in N if the range is already sorted, and NlogN otherwise (where N is distance(__first,__last)).

set() [5/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            const set< _Key, _Compare, _Alloc > & )  [default]
```

Set copy constructor.
Whether the allocator is copied depends on the allocator traits.

set() [6/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            set< _Key, _Compare, _Alloc > && )  [default]
```

Set move constructor
The newly-created set contains the exact contents of the moved instance. The moved instance is a valid, but unspecified, set.

set() [7/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            initializer_list< value_type > __l,
            const _Compare & __comp = _Compare(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds a set from an initializer_list.

Parameters

__l	An initializer_list.
__comp	A comparison functor.
__a	An allocator object.

Create a set consisting of copies of the elements in the list. This is linear in N if the list is already sorted, and NlogN

otherwise (where N is *__l.size()*).

set() [8/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            const allocator_type & __a)  [inline], [explicit]
```

Allocator-extended default constructor.

set() [9/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            const set< _Key, _Compare, _Alloc > & __x,
            const __type_identity_t< allocator_type > & __a)  [inline]
```

Allocator-extended copy constructor.

set() [10/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            set< _Key, _Compare, _Alloc > && __x,
            const __type_identity_t< allocator_type > & __a)  [inline], [noexcept]
```

Allocator-extended move constructor.

set() [11/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::set (
            initializer_list< value_type > __l,
            const allocator_type & __a)  [inline]
```

Allocator-extended initializer-list constructor.

set() [12/12]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
std::set< _Key, _Compare, _Alloc >::set (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a)  [inline]
```

Allocator-extended range constructor.

~set()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::set< _Key, _Compare, _Alloc >::~set ()  [default]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

6.912.4 Member Function Documentation

begin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::begin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

cbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::cbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the set. Iteration is done in ascending order according to the keys.

cend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

clear()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
void std::set< _Key, _Compare, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in a set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _Kt>
auto std::set< _Key, _Compare, _Alloc >::contains (
            const _Kt & __x) const -> decltype(_M_t._M_find_tr(__x), void(), true)  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _*x*	Key of elements to be located.

Returns

True if there is an element with the specified key.

contains() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
bool std::set< _Key, _Compare, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is an element with the specified key.

count()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Element to located.

Returns

Number of elements with specified key.

This function only makes sense for multisets; for set the result will either be 0 (not present) or 1 (present).

crbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::crbegin () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

crend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::crend () const  [inline], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

emplace()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename... _Args>
std::pair< iterator, bool > std::set< _Key, _Compare, _Alloc >::emplace (
            _Args &&... __args) [inline]
```

Attempts to build and insert an element into the set.

Parameters

__args	Arguments used to generate an element.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.
Insertion requires logarithmic time.

emplace_hint()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename... _Args>
iterator std::set< _Key, _Compare, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&... __args) [inline]
```

Attempts to insert an element into the set.

Parameters

__pos	An iterator that serves as a hint as to where the element should be inserted.
__args	Arguments used to generate the element to be inserted.

Returns

An iterator that points to the element with key equivalent to the one generated from *__args* (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↩html#containers.associative.insert_hints
Insertion requires logarithmic time (if the hint is not taken).

empty()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
bool std::set< _Key, _Compare, _Alloc >::empty () const [inline], [nodiscard], [noexcept]
```

Returns true if the set is empty.

end()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the set. Iteration is done in ascending order according to the keys.

equal_range() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::pair< iterator, iterator > std::set< _Key, _Compare, _Alloc >::equal_range (
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multisets.

equal_range() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::pair< const_iterator, const_iterator > std::set< _Key, _Compare, _Alloc >::equal_range (
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function is equivalent to

```
std::make_pair(c.lower_bound(val),
               c.upper_bound(val))
```

(but is faster than making the calls separately).
This function probably only makes sense for multisets.

erase() [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [__first,__last) range of elements from a set.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from a set.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from a set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in a set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
const_iterator std::set< _Key, _Compare, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in a set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
allocator_type std::set< _Key, _Compare, _Alloc >::get_allocator () const  [inline], [noexcept]
```

Returns the allocator object with which the set was constructed.

insert() [1/5]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
template<typename _InputIterator>
void std::set< _Key, _Compare, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/5]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
std::pair< iterator, bool > std::set< _Key, _Compare, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Attempts to insert an element into the set.

Parameters

_↩ *_x*	Element to be inserted.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the set. A set relies on unique keys and thus an element is only inserted if it is not already present in the set.
Insertion requires logarithmic time.
Referenced by std::set< _Key, _Cmp, polymorphic_allocator< _Key > >::insert().

insert() [3/5]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::insert (
            const_iterator __position,
            const value_type & __x)  [inline]
```

Attempts to insert an element into the set.

Parameters

__position	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the element with key of *__x* (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
If a heterogeneous key __k matches a range of elements, an iterator to the first is returned.
Insertion requires logarithmic time (if the hint is not taken).

insert() [4/5]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
iterator std::set< _Key, _Compare, _Alloc >::insert (
            const_iterator __position,
            value_type && __x)  [inline]
```

Attempts to insert an element into the set.

Parameters

__position	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the element with key of *__x* (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
If a heterogeneous key __k matches a range of elements, an iterator to the first is returned.
Insertion requires logarithmic time (if the hint is not taken).

insert() [5/5]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<_Key>>
void std::set< _Key, _Compare, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Attempts to insert a list of elements into the set.

Parameters

↩ _↩ ↩ _↩ *l*	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

key_comp()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
key_compare std::set< _Key, _Compare, _Alloc >::key_comp () const  [inline]
```

Returns the comparison object with which the set was constructed.

lower_bound() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::lower_bound (
            const key_type & __x)  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ *_x*	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

lower_bound() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
const_iterator std::set< _Key, _Compare, _Alloc >::lower_bound (
            const key_type & __x) const  [inline]
```

Finds the beginning of a subsequence matching given key.

Parameters

_↩ *_x*	Key to be located.

Returns

Iterator pointing to first element equal to or greater than key, or end().

This function returns the first element of a subsequence of elements that matches the given key. If unsuccessful it returns an iterator pointing to the first element that has a greater value than given key or end() if no such element exists.

max_size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the maximum size of the set.

operator=() [1/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
set & std::set< _Key, _Compare, _Alloc >::operator= (
            const set< _Key, _Compare, _Alloc > & )  [default]
```

Set assignment operator.
Whether the allocator is copied depends on the allocator traits.

operator=() [2/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
set & std::set< _Key, _Compare, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Set list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a set with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the set and that the resulting set's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
set & std::set< _Key, _Compare, _Alloc >::operator= (
            set< _Key, _Compare, _Alloc > && )  [default]
```

Move assignment operator.

rbegin()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
```

```
reverse_iterator std::set< _Key, _Compare, _Alloc >::rbegin () const  [inline], [noexcept]
```
Returns a read-only (constant) iterator that points to the last element in the set. Iteration is done in descending order according to the keys.

rend()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
reverse_iterator std::set< _Key, _Compare, _Alloc >::rend () const  [inline], [noexcept]
```
Returns a read-only (constant) reverse iterator that points to the last pair in the set. Iteration is done in descending order according to the keys.

size()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
size_type std::set< _Key, _Compare, _Alloc >::size () const  [inline], [noexcept]
```
Returns the size of the set.

swap()

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
void std::set< _Key, _Compare, _Alloc >::swap (
            set< _Key, _Compare, _Alloc > & __x)  [inline], [noexcept]
```
Swaps data with another set.

Parameters

_↩ _x	A set of the same element and allocator types.

This exchanges the elements between two sets in constant time. (It is only swapping a pointer, an integer, and an instance of the `Compare` type (which itself is often stateless and empty), so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.
Whether the allocators are swapped depends on the allocator traits.
Referenced by std::swap().

upper_bound() [1/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
iterator std::set< _Key, _Compare, _Alloc >::upper_bound (
            const key_type & __x)  [inline]
```
Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

upper_bound() [2/2]

```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
const_iterator std::set< _Key, _Compare, _Alloc >::upper_bound (
            const key_type & __x) const  [inline]
```

Finds the end of a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to the first element greater than key, or end().

value_comp()

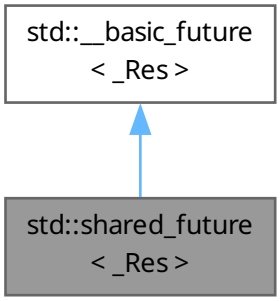
```
template<typename _Key, typename _Compare = std::less<_Key>, typename _Alloc = std::allocator<↩
_Key>>
value_compare std::set< _Key, _Compare, _Alloc >::value_comp () const  [inline]
```

Returns the comparison object with which the set was constructed.
The documentation for this class was generated from the following files:

- stl_multiset.h
- stl_set.h

6.913 std::shared_future< _Res > Class Template Reference

`#include <future>`
Inheritance diagram for std::shared_future< _Res >:



Public Member Functions

- shared_future (const shared_future &__sf) noexcept
- shared_future (future< _Res > &&__uf) noexcept
- shared_future (shared_future &&__sf) noexcept
- const _Res & get () const
- shared_future & **operator=** (const shared_future &__sf) noexcept
- shared_future & **operator=** (shared_future &&__sf) noexcept
- bool **valid** () const noexcept
- void **wait** () const
- template<typename _Rep, typename _Period>
  future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- template<typename _Clock, typename _Duration>
  future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< _Res > & **__result_type**
- typedef shared_ptr< _State_base > **__state_type**

Protected Member Functions

- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

6.913.1 Detailed Description

template<typename _Res>
class std::shared_future< _Res >

Primary template for shared_future.

6.913.2 Constructor & Destructor Documentation

shared_future() [1/3]

```
template<typename _Res>
std::shared_future< _Res >::shared_future (
            const shared_future< _Res > & __sf)  [inline], [noexcept]
```

Copy constructor.

shared_future() [2/3]

```
template<typename _Res>
std::shared_future< _Res >::shared_future (
            future< _Res > && __uf)  [inline], [noexcept]
```

Construct from a future rvalue.
References std::move().

shared_future() [3/3]

```
template<typename _Res>
std::shared_future< _Res >::shared_future (
            shared_future< _Res > && __sf)  [inline], [noexcept]
```

Construct from a shared_future rvalue.
References std::move().

6.913.3 Member Function Documentation

_M_get_result()

```
template<typename _Res>
__result_type std::__basic_future< _Res >::_M_get_result () const  [inline], [protected], [inherited]
```

Wait for the state to be ready and rethrow any stored exception.
Referenced by std::future< _Res >::get(), and std::shared_future< _Res >::get().

get()

```
template<typename _Res>
const _Res & std::shared_future< _Res >::get () const  [inline]
```

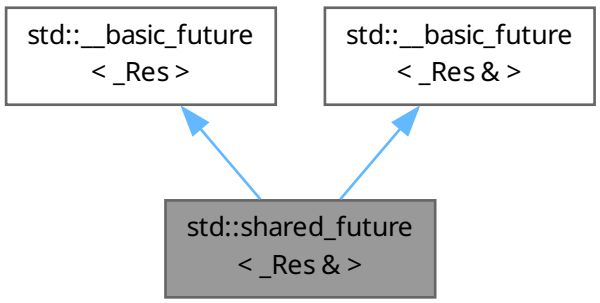
Retrieving the value.
References std::__basic_future< _Res >::_M_get_result().
The documentation for this class was generated from the following file:

- future

6.914 std::shared_future< _Res & > Class Template Reference

```
#include <future>
```

Inheritance diagram for std::shared_future< _Res & >:



Public Member Functions

- shared_future (const shared_future &__sf)
- shared_future (future< _Res & > &&__uf) noexcept
- shared_future (shared_future &&__sf) noexcept
- _Res & get () const
- shared_future & **operator=** (const shared_future &__sf)
- shared_future & **operator=** (shared_future &&__sf) noexcept
- bool **valid** () const noexcept
- void **wait** () const
- future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< _Res > & **__result_type**
- typedef shared_ptr< _State_base > **__state_type**

Protected Member Functions

- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

6.914.1 Detailed Description

template<typename _Res>
class std::shared_future< _Res & >

Partial specialization for shared_future<R&>.

6.914.2 Constructor & Destructor Documentation

shared_future() [1/3]

```
template<typename _Res>
std::shared_future< _Res & >::shared_future (
            const shared_future< _Res & > & __sf)  [inline]
```

Copy constructor.

shared_future() [2/3]

```
template<typename _Res>
std::shared_future< _Res & >::shared_future (
            future< _Res & > && __uf)  [inline], [noexcept]
```

Construct from a future rvalue.
References std::move().

shared_future() [3/3]

```
template<typename _Res>
std::shared_future< _Res & >::shared_future (
            shared_future< _Res & > && __sf)  [inline], [noexcept]
```

Construct from a shared_future rvalue.
References std::move().

6.914.3 Member Function Documentation

_M_get_result()

```
__result_type std::__basic_future< _Res >::_M_get_result () const  [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.
Referenced by get().

get()

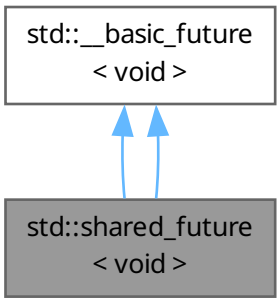
```
template<typename _Res>
_Res & std::shared_future< _Res & >::get () const  [inline]
```

Retrieving the value.
References _M_get_result().
The documentation for this class was generated from the following file:

- future

6.915 std::shared_future< void > Class Reference

`#include <future>`
Inheritance diagram for std::shared_future< void >:



Public Member Functions

- shared_future (const shared_future &__sf)
- shared_future (future< void > &&__uf) noexcept
- shared_future (shared_future &&__sf) noexcept
- void **get** () const
- shared_future & **operator=** (const shared_future &__sf)
- shared_future & **operator=** (shared_future &&__sf) noexcept
- bool **valid** () const noexcept
- void **wait** () const
- future_status **wait_for** (const chrono::duration< _Rep, _Period > &__rel) const
- future_status **wait_until** (const chrono::time_point< _Clock, _Duration > &__abs) const

Protected Types

- typedef __future_base::_Result< void > & **__result_type**
- typedef shared_ptr< _State_base > **__state_type**

Protected Member Functions

- __result_type _M_get_result () const
- void **_M_swap** (__basic_future &__that) noexcept

6.915.1 Detailed Description

Explicit specialization for shared_future<void>.

6.915.2 Constructor & Destructor Documentation

shared_future() [1/3]

```
std::shared_future< void >::shared_future (
            const shared_future< void > & __sf)  [inline]
```

Copy constructor.

shared_future() [2/3]

```
std::shared_future< void >::shared_future (
            future< void > && __uf)  [inline], [noexcept]
```

Construct from a future rvalue.
References std::move().

shared_future() [3/3]

```
std::shared_future< void >::shared_future (
            shared_future< void > && __sf)  [inline], [noexcept]
```

Construct from a shared_future rvalue.
References std::move().

6.915.3 Member Function Documentation

_M_get_result()

```
__result_type std::__basic_future< void >::_M_get_result () const  [inline], [protected]
```

Wait for the state to be ready and rethrow any stored exception.
The documentation for this class was generated from the following file:

- future

6.916 std::shared_lock< _Mutex > Class Template Reference

```
#include <shared_mutex>
```

Public Types

- typedef _Mutex **mutex_type**

Public Member Functions

- **shared_lock** (mutex_type &__m)
- **shared_lock** (mutex_type &__m, adopt_lock_t)
- template<typename _Rep, typename _Period>
  shared_lock (mutex_type &__m, const chrono::duration< _Rep, _Period > &__rel_time)
- template<typename _Clock, typename _Duration>
  shared_lock (mutex_type &__m, const chrono::time_point< _Clock, _Duration > &__abs_time)
- **shared_lock** (mutex_type &__m, defer_lock_t) noexcept
- **shared_lock** (mutex_type &__m, try_to_lock_t)
- **shared_lock** (shared_lock &&__sl) noexcept
- **shared_lock** (shared_lock const &)=delete
- void **lock** ()
- mutex_type * **mutex** () const noexcept
- **operator bool** () const noexcept
- shared_lock & **operator=** (shared_lock &&__sl) noexcept

- shared_lock & **operator=** (shared_lock const &)=delete
- bool **owns_lock** () const noexcept
- mutex_type * **release** () noexcept
- void **swap** (shared_lock &__u) noexcept
- bool **try_lock** ()
- template<typename _Rep, typename _Period>
  bool **try_lock_for** (const chrono::duration< _Rep, _Period > &__rel_time)
- template<typename _Clock, typename _Duration>
  bool **try_lock_until** (const chrono::time_point< _Clock, _Duration > &__abs_time)
- void **unlock** ()

6.916.1 Detailed Description

template<typename _Mutex>
class std::shared_lock< _Mutex >

shared_lock
The documentation for this class was generated from the following file:

- shared_mutex

6.917 std::shared_ptr< _Tp > Class Template Reference

`#include <memory>`

Public Types

- using element_type
- using **weak_type**

Public Member Functions

- constexpr shared_ptr () noexcept
- template<typename _Yp, typename = _Constructible<_Yp*>>
  shared_ptr (_Yp *__p)
- template<typename _Yp, typename _Deleter, typename = _Constructible<_Yp*, _Deleter>>
  shared_ptr (_Yp *__p, _Deleter __d)
- template<typename _Yp, typename _Deleter, typename _Alloc, typename = _Constructible<_Yp*, _Deleter, _Alloc>>
  shared_ptr (_Yp *__p, _Deleter __d, _Alloc __a)
- template<typename _Yp, typename = _Constructible<auto_ptr<_Yp>>>
  shared_ptr (auto_ptr< _Yp > &&__r)
- shared_ptr (const shared_ptr &) noexcept=default
- template<typename _Yp, typename = _Constructible<const shared_ptr<_Yp>&>>
  shared_ptr (const shared_ptr< _Yp > &__r) noexcept
- template<typename _Yp>
  shared_ptr (const shared_ptr< _Yp > &__r, element_type *__p) noexcept
- template<typename _Yp, typename = _Constructible<const weak_ptr<_Yp>&>>
  shared_ptr (const weak_ptr< _Yp > &__r)
- template<typename _Deleter>
  shared_ptr (nullptr_t __p, _Deleter __d)
- template<typename _Deleter, typename _Alloc>
  shared_ptr (nullptr_t __p, _Deleter __d, _Alloc __a)
- constexpr shared_ptr (nullptr_t) noexcept
- shared_ptr (shared_ptr &&__r) noexcept

- template<typename _Yp, typename = _Constructible<shared_ptr<_Yp>>>
  shared_ptr (shared_ptr< _Yp > &&__r) noexcept
- template<typename _Yp>
  shared_ptr (shared_ptr< _Yp > &&__r, element_type *__p) noexcept
- template<typename _Tp1, typename>
  shared_ptr (std::auto_ptr< _Tp1 > &&__r)
- template<typename _Yp, typename _Del, typename = _Constructible<unique_ptr<_Yp, _Del>>>
  shared_ptr (unique_ptr< _Yp, _Del > &&__r)
- element_type * get () const noexcept
- operator bool () const noexcept
- element_type & **operator*** () const noexcept
- element_type * **operator->** () const noexcept
- template<typename _Yp>
  _Assignable< auto_ptr< _Yp > > **operator=** (auto_ptr< _Yp > &&__r)
- shared_ptr & **operator=** (const shared_ptr &) noexcept=default
- template<typename _Yp>
  _Assignable< const shared_ptr< _Yp > & > **operator=** (const shared_ptr< _Yp > &__r) noexcept
- shared_ptr & **operator=** (shared_ptr &&__r) noexcept
- template<class _Yp>
  _Assignable< shared_ptr< _Yp > > **operator=** (shared_ptr< _Yp > &&__r) noexcept
- template<typename _Yp, typename _Del>
  _Assignable< unique_ptr< _Yp, _Del > > **operator=** (unique_ptr< _Yp, _Del > &&__r)
- void **reset** () noexcept
- template<typename _Yp>
  _SafeConv< _Yp > **reset** (_Yp *__p)
- template<typename _Yp, typename _Deleter>
  _SafeConv< _Yp > **reset** (_Yp *__p, _Deleter __d)
- template<typename _Yp, typename _Deleter, typename _Alloc>
  _SafeConv< _Yp > **reset** (_Yp *__p, _Deleter __d, _Alloc __a)
- void swap (__shared_ptr< _Tp, _Lp > &__other) noexcept
- bool unique () const noexcept
- long use_count () const noexcept

- template<typename _Tp1>
  bool owner_before (__shared_ptr< _Tp1, _Lp > const &__rhs) const noexcept
- template<typename _Tp1>
  bool owner_before (__weak_ptr< _Tp1, _Lp > const &__rhs) const noexcept

Friends

- template<typename _Yp, typename _Alloc, typename... _Args>
  shared_ptr< _NonArray< _Yp > > **allocate_shared** (const _Alloc &, _Args &&...)
- template<typename _Yp, typename... _Args>
  shared_ptr< _NonArray< _Yp > > **make_shared** (_Args &&...)
- class **weak_ptr**< **_Tp** >

Related Symbols

(Note that these are not member symbols.)

- template<typename _Del, typename _Tp>
  _Del * get_deleter (const shared_ptr< _Tp > &__p) noexcept

- template<typename _Ch, typename _Tr, typename _Tp, _Lock_policy _Lp>
  std::basic_ostream< _Ch, _Tr > & operator<< (std::basic_ostream< _Ch, _Tr > &__os, const __shared_ptr< _Tp, _Lp > &__p)

- template<typename _Tp, typename _Up>
  bool operator== (const shared_ptr< _Tp > &__a, const shared_ptr< _Up > &__b) noexcept
- template<typename _Tp>
  bool operator== (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp>
  void swap (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > static_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > const_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > dynamic_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > reinterpret_pointer_cast (const shared_ptr< _Up > &__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > static_pointer_cast (shared_ptr< _Up > &&__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > const_pointer_cast (shared_ptr< _Up > &&__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > dynamic_pointer_cast (shared_ptr< _Up > &&__r) noexcept
- template<typename _Tp, typename _Up>
  shared_ptr< _Tp > reinterpret_pointer_cast (shared_ptr< _Up > &&__r) noexcept

- template<typename _Tp, typename _Alloc, typename... _Args>
  shared_ptr< _NonArray< _Tp > > allocate_shared (const _Alloc &__a, _Args &&... __args)
- template<typename _Tp, typename... _Args>
  shared_ptr< _NonArray< _Tp > > make_shared (_Args &&... __args)

6.917.1 Detailed Description

template<typename _Tp>
class std::shared_ptr< _Tp >

A smart pointer with reference-counted copy semantics.

Since

C++11

A `shared_ptr` object is either empty or *owns* a pointer passed to the constructor. Copies of a `shared_ptr` share ownership of the same pointer. When the last `shared_ptr` that owns the pointer is destroyed or reset, the owned pointer is freed (either by `delete` or by invoking a custom deleter that was passed to the constructor).
A `shared_ptr` also stores another pointer, which is usually (but not always) the same pointer as it owns. The stored pointer can be retrieved by calling the `get()` member function.
The equality and relational operators for `shared_ptr` only compare the stored pointer returned by `get()`, not the owned pointer. To test whether two `shared_ptr` objects share ownership of the same pointer see `std::shared_ptr::owner_before` and `std::owner_less`.

6.917.2 Member Typedef Documentation

element_type

```
template<typename _Tp>
using std::shared_ptr< _Tp >::element_type
```

The type pointed to by the stored pointer, remove_extent_t<_Tp>.

6.917.3 Constructor & Destructor Documentation

shared_ptr() [1/14]

```
template<typename _Tp>
std::shared_ptr< _Tp >::shared_ptr ()  [inline], [constexpr], [noexcept]
```

Construct an empty shared_ptr.

Postcondition

use_count()==0 && get()==0

shared_ptr() [2/14]

```
template<typename _Tp>
std::shared_ptr< _Tp >::shared_ptr (
            const shared_ptr< _Tp > & )  [default], [noexcept]
```

Copy constructor.

shared_ptr() [3/14]

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<_Yp*>>
std::shared_ptr< _Tp >::shared_ptr (
            _Yp * __p)  [inline], [explicit]
```

Construct a shared_ptr that owns the pointer *__p*.

Parameters

_↩ _p	A pointer that is convertible to element_type*.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case `delete` *__p* is called.

shared_ptr() [4/14]

```
template<typename _Tp>
template<typename _Yp, typename _Deleter, typename = _Constructible<_Yp*, _Deleter>>
std::shared_ptr< _Tp >::shared_ptr (
```

```
            _Yp * __p,
            _Deleter __d)  [inline]
```

Construct a shared_ptr that owns the pointer __p and the deleter __d.

Parameters

_↩ _p	A pointer.
_↩ _d	A deleter.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case *__d(__p)* is called.

Requirements: _Deleter's copy constructor and destructor must not throw
__shared_ptr will release __p by calling __d(__p)

shared_ptr() [5/14]

```
template<typename _Tp>
template<typename _Deleter>
std::shared_ptr< _Tp >::shared_ptr (
            nullptr_t __p,
            _Deleter __d)  [inline]
```

Construct a shared_ptr that owns a null pointer and the deleter __d.

Parameters

_↩ _p	A null pointer constant.
_↩ _d	A deleter.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case *__d(__p)* is called.

Requirements: _Deleter's copy constructor and destructor must not throw
The last owner will call __d(__p)

shared_ptr() [6/14]

```
template<typename _Tp>
template<typename _Yp, typename _Deleter, typename _Alloc, typename = _Constructible<_Yp*, _↩
Deleter, _Alloc>>
std::shared_ptr< _Tp >::shared_ptr (
            _Yp * __p,
            _Deleter __d,
            _Alloc __a)  [inline]
```

Construct a shared_ptr that owns the pointer *__p* and the deleter *__d*.

Parameters

_↩ _*p*	A pointer.
_↩ _*d*	A deleter.
_↩ _*a*	An allocator.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case *__d(__p)* is called.

Requirements: _Deleter's copy constructor and destructor must not throw _Alloc's copy constructor and destructor must not throw.
__shared_ptr will release __p by calling __d(__p)

shared_ptr() [7/14]

```
template<typename _Tp>
template<typename _Deleter, typename _Alloc>
std::shared_ptr< _Tp >::shared_ptr (
            nullptr_t __p,
            _Deleter __d,
            _Alloc __a)  [inline]
```

Construct a shared_ptr that owns a null pointer and the deleter *__d*.

Parameters

_↩ _*p*	A null pointer constant.
_↩ _*d*	A deleter.
_↩ _*a*	An allocator.

Postcondition

use_count() == 1 && get() == __p

Exceptions

std::bad_alloc,in	which case *__d(__p)* is called.

Requirements: _Deleter's copy constructor and destructor must not throw _Alloc's copy constructor and destructor must not throw.
The last owner will call __d(__p)

shared_ptr() [8/14]

```
template<typename _Tp>
template<typename _Yp>
std::shared_ptr< _Tp >::shared_ptr (
            const shared_ptr< _Yp > & __r,
            element_type * __p)  [inline], [noexcept]
```

Constructs a `shared_ptr` instance that stores `__p` and shares ownership with `__r`.

Parameters

__r	A `shared_ptr`.
__p	A pointer that will remain valid while `*__r` is valid.

Postcondition

`get() == __p && use_count() == __r.use_count()`

This can be used to construct a `shared_ptr` to a sub-object of an object managed by an existing `shared_ptr`. The complete object will remain valid while any `shared_ptr` owns it, even if they don't store a pointer to the complete object.

```
shared_ptr<pair<int,int>> pii(new pair<int,int>());
shared_ptr<int> pi(pii, &pii->first);
assert(pii.use_count() == 2);
```

shared_ptr() [9/14]

```
template<typename _Tp>
template<typename _Yp>
std::shared_ptr< _Tp >::shared_ptr (
            shared_ptr< _Yp > && __r,
            element_type * __p)  [inline], [noexcept]
```

Constructs a `shared_ptr` instance that stores `__p` and shares ownership with `__r`.

Parameters

_↩ _r	A shared_ptr.
_↩ _p	A pointer that will remain valid while *__r is valid.

Postcondition

```
get() == __p && !__r.use_count() && !__r.get()
```

Since

C++17

This can be used to construct a shared_ptr to a sub-object of an object managed by an existing shared_ptr. The complete object will remain valid while any shared_ptr owns it, even if they don't store a pointer to the complete object.

```
shared_ptr<pair<int,int>> pii(new pair<int,int>());
shared_ptr<int> pi1(pii, &pii->first);
assert(pii.use_count() == 2);
shared_ptr<int> pi2(std::move(pii), &pii->second);
assert(pii.use_count() == 0);
```

shared_ptr() [10/14]

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<const shared_ptr<_Yp>&>>
std::shared_ptr< _Tp >::shared_ptr (
            const shared_ptr< _Yp > & __r)  [inline], [noexcept]
```

If __r is empty, constructs an empty shared_ptr; otherwise construct a shared_ptr that shares ownership with __r.

Parameters

↩ _↩ ↩ _↩ r	A shared_ptr.

Postcondition

get() == __r.get() && use_count() == __r.use_count()

shared_ptr() [11/14]

```
template<typename _Tp>
std::shared_ptr< _Tp >::shared_ptr (
            shared_ptr< _Tp > && __r)  [inline], [noexcept]
```

Move-constructs a shared_ptr instance from __r.

Parameters

↩ _↩ ↩ _↩ *r*	A shared_ptr rvalue.

Postcondition

∗this contains the old value of *__r*, *__r* is empty.

shared_ptr() [12/14]

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<shared_ptr<_Yp>>>
std::shared_ptr< _Tp >::shared_ptr (
            shared_ptr< _Yp > && __r)  [inline], [noexcept]
```

Move-constructs a shared_ptr instance from *__r*.

Parameters

↩ _↩ ↩ _↩ *r*	A shared_ptr rvalue.

Postcondition

∗this contains the old value of *__r*, *__r* is empty.

shared_ptr() [13/14]

```
template<typename _Tp>
template<typename _Yp, typename = _Constructible<const weak_ptr<_Yp>&>>
std::shared_ptr< _Tp >::shared_ptr (
            const weak_ptr< _Yp > & __r)  [inline], [explicit]
```

Constructs a shared_ptr that shares ownership with *__r* and stores a copy of the pointer stored in *__r*.

Parameters

↩ _↩ ↩ _↩ *r*	A weak_ptr.

Postcondition

use_count() == __r.use_count()

Exceptions

bad_weak_ptr	when __r.expired(), in which case the constructor has no effect.

shared_ptr() [14/14]

```
template<typename _Tp>
std::shared_ptr< _Tp >::shared_ptr (
            nullptr_t )  [inline], [constexpr], [noexcept]
```

Construct an empty shared_ptr.

Postcondition

use_count() == 0 && get() == nullptr

6.917.4 Member Function Documentation

get()

```
template<typename _Tp, _Lock_policy _Lp>
element_type * std::__shared_ptr< _Tp, _Lp >::get () const  [inline], [noexcept], [inherited]
```

Return the stored pointer.

operator bool()

```
template<typename _Tp, _Lock_policy _Lp>
std::__shared_ptr< _Tp, _Lp >::operator bool () const  [inline], [explicit], [noexcept], [inherited]
```

Return true if the stored pointer is not null.

owner_before() [1/2]

```
template<typename _Tp, _Lock_policy _Lp>
template<typename _Tp1>
bool std::__shared_ptr< _Tp, _Lp >::owner_before (
            __shared_ptr< _Tp1, _Lp > const & __rhs) const  [inline], [noexcept], [inherited]
```

Define an ordering based on ownership.

This function defines a strict weak ordering between two shared_ptr or weak_ptr objects, such that one object is less than the other unless they share ownership of the same pointer, or are both empty.

owner_before() [2/2]

```
template<typename _Tp, _Lock_policy _Lp>
template<typename _Tp1>
bool std::__shared_ptr< _Tp, _Lp >::owner_before (
            __weak_ptr< _Tp1, _Lp > const & __rhs) const  [inline], [noexcept], [inherited]
```

Define an ordering based on ownership.

This function defines a strict weak ordering between two shared_ptr or weak_ptr objects, such that one object is less than the other unless they share ownership of the same pointer, or are both empty.

swap()

```
template<typename _Tp, _Lock_policy _Lp>
void std::__shared_ptr< _Tp, _Lp >::swap (
             __shared_ptr< _Tp, _Lp > & __other)  [inline], [noexcept], [inherited]
```

Exchange both the owned pointer and the stored pointer.

unique()

```
template<typename _Tp, _Lock_policy _Lp>
bool std::__shared_ptr< _Tp, _Lp >::unique () const  [inline], [noexcept], [inherited]
```

Return true if use_count() == 1.

use_count()

```
template<typename _Tp, _Lock_policy _Lp>
long std::__shared_ptr< _Tp, _Lp >::use_count () const  [inline], [noexcept], [inherited]
```

If ∗this owns a pointer, return the number of owners, otherwise zero.
The documentation for this class was generated from the following files:

- bits/shared_ptr.h
- auto_ptr.h

6.918 std::shared_timed_mutex Class Reference

```
#include <shared_mutex>
```

Public Member Functions

- **shared_timed_mutex** (const shared_timed_mutex &)=delete
- void **lock** ()
- void **lock_shared** ()
- shared_timed_mutex & **operator=** (const shared_timed_mutex &)=delete
- bool **try_lock** ()
- template<typename _Rep, typename _Period>
  bool **try_lock_for** (const chrono::duration< _Rep, _Period > &__rtime)
- bool **try_lock_shared** ()
- template<typename _Rep, typename _Period>
  bool **try_lock_shared_for** (const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock, typename _Duration>
  bool **try_lock_shared_until** (const chrono::time_point< _Clock, _Duration > &__abs_time)
- template<typename _Clock, typename _Duration>
  bool **try_lock_until** (const chrono::time_point< _Clock, _Duration > &__abs_time)
- void **unlock** ()
- void **unlock_shared** ()

6.918.1 Detailed Description

The standard shared timed mutex type.
The documentation for this class was generated from the following file:

- shared_mutex

6.919 std::shuffle_order_engine< _RandomNumberEngine, __k > Class Template Reference

`#include <random>`

Public Types

- template<typename _Sseq>
  using **_If_seed_seq**
- typedef _RandomNumberEngine::result_type result_type

Public Member Functions

- shuffle_order_engine ()
- shuffle_order_engine (_RandomNumberEngine &&__rng)
- template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
  shuffle_order_engine (_Sseq &__q)
- shuffle_order_engine (const _RandomNumberEngine &__rng)
- shuffle_order_engine (result_type __s)
- const _RandomNumberEngine & base () const noexcept
- void discard (unsigned long long __z)
- result_type operator() ()
- void seed ()
- template<typename _Sseq>
  _If_seed_seq< _Sseq > seed (_Sseq &__q)
- void seed (result_type __s)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr size_t **table_size**

Friends

- template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &__x)
- bool operator== (const shuffle_order_engine &__lhs, const shuffle_order_engine &__rhs)
- template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::shuffle_order_engine< _RandomNumberEngine1, __k1 > &__x)

6.919.1 Detailed Description

template<typename _RandomNumberEngine, size_t __k>
class std::shuffle_order_engine< _RandomNumberEngine, __k >

Produces random numbers by reordering random numbers from some base engine.
The values from the base engine are stored in a sequence of size `__k` and shuffled by an algorithm that depends on those values.

Since

C++11

6.919.2 Member Typedef Documentation

result_type

```
template<typename _RandomNumberEngine, size_t __k>
typedef _RandomNumberEngine::result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::result_type
```

The type of the generated random value.

6.919.3 Constructor & Destructor Documentation

shuffle_order_engine() [1/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine ()  [inline]
```

Constructs a default shuffle_order_engine engine.
The underlying engine is default constructed as well.

shuffle_order_engine() [2/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
            const _RandomNumberEngine & __rng)  [inline], [explicit]
```

Copy constructs a shuffle_order_engine engine.
Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

shuffle_order_engine() [3/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
            _RandomNumberEngine && __rng)  [inline], [explicit]
```

Move constructs a shuffle_order_engine engine.
Copies an existing base class random number generator.

Parameters

__rng	An existing (base class) engine object.

shuffle_order_engine() [4/5]

```
template<typename _RandomNumberEngine, size_t __k>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
            result_type __s)  [inline], [explicit]
```

Seed constructs a shuffle_order_engine engine.
Constructs the underlying generator engine seeded with __s.

Parameters

_↩ _s	A seed value for the base class engine.

shuffle_order_engine() [5/5]

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::shuffle_order_engine< _RandomNumberEngine, __k >::shuffle_order_engine (
            _Sseq & __q)  [inline], [explicit]
```

Generator construct a shuffle_order_engine engine.

Parameters

_↩ _q	A seed sequence.

6.919.4 Member Function Documentation

base()

```
template<typename _RandomNumberEngine, size_t __k>
const _RandomNumberEngine & std::shuffle_order_engine< _RandomNumberEngine, __k >::base () const
[inline], [noexcept]
```

Gets a const reference to the underlying generator engine object.

discard()

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::discard (
            unsigned long long __z)  [inline]
```

Discard a sequence of random numbers.

max()

```
template<typename _RandomNumberEngine, size_t __k>
constexpr result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::max ()  [inline],
[static], [constexpr]
```

Gets the maximum value in the generated random number range.

min()

```
template<typename _RandomNumberEngine, size_t __k>
constexpr result_type std::shuffle_order_engine< _RandomNumberEngine, __k >::min ()  [inline],
[static], [constexpr]
```

Gets the minimum value in the generated random number range.

operator()()

```
template<typename _RandomNumberEngine, size_t __k>
shuffle_order_engine< _RandomNumberEngine, __k >::result_type std::shuffle_order_engine< _↩
RandomNumberEngine, __k >::operator() ()
```

Gets the next value in the generated random number sequence.
References std::max(), and std::min().

seed() [1/3]

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed ()  [inline]
```

Reseeds the shuffle_order_engine object with the default seed for the underlying base class generator engine.

seed() [2/3]

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _Sseq>
_If_seed_seq< _Sseq > std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (
            _Sseq & __q)  [inline]
```

Reseeds the shuffle_order_engine object with the given seed sequence.

Parameters

_↩ _q	A seed generator function.

seed() [3/3]

```
template<typename _RandomNumberEngine, size_t __k>
void std::shuffle_order_engine< _RandomNumberEngine, __k >::seed (
            result_type __s)  [inline]
```

Reseeds the shuffle_order_engine object with the default seed for the underlying base class generator engine.

6.919.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & __x)  [friend]
```

Inserts the current state of a shuffle_order_engine random number generator engine __x into the output stream __os.

Parameters

__os	An output stream.
__x	A shuffle_order_engine random number generator engine.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RandomNumberEngine, size_t __k>
bool operator== (
             const shuffle_order_engine< _RandomNumberEngine, __k > & __lhs,
             const shuffle_order_engine< _RandomNumberEngine, __k > & __rhs)  [friend]
```

Compares two shuffle_order_engine random number generator objects of the same type for equality.

Parameters

__lhs	A shuffle_order_engine random number generator object.
__rhs	Another shuffle_order_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

operator>>

```
template<typename _RandomNumberEngine, size_t __k>
template<typename _RandomNumberEngine1, size_t __k1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
             std::basic_istream< _CharT, _Traits > & __is,
             std::shuffle_order_engine< _RandomNumberEngine1, __k1 > & __x)  [friend]
```

Extracts the current state of a % subtract_with_carry_engine random number generator engine __x from the input stream __is.

Parameters

__is	An input stream.
__x	A shuffle_order_engine random number generator engine.

Returns

The input stream with the state of __x extracted or in an error state.

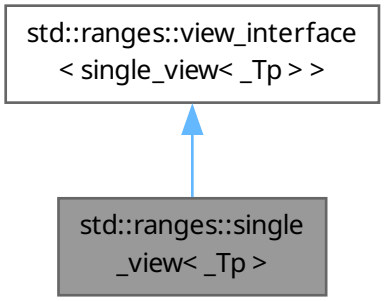
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.920 std::ranges::single_view< _Tp > Class Template Reference

```
#include <ranges>
```

Inheritance diagram for std::ranges::single_view< _Tp >:



Public Member Functions

- constexpr **single_view** (_Tp &&__t) noexcept(is_nothrow_move_constructible_v< _Tp >)
- constexpr **single_view** (const _Tp &__t) noexcept(is_nothrow_copy_constructible_v< _Tp >)
- template<typename... _Args>
  constexpr **single_view** (in_place_t, _Args &&... __args) noexcept(is_nothrow_constructible_v< _Tp, _Args... >)
- constexpr decltype(auto) **back** ()
- constexpr const _Tp * **begin** () const noexcept
- constexpr _Tp * **begin** () noexcept
- constexpr auto **cbegin** ()
- constexpr auto **cend** ()
- constexpr const _Tp * **data** () const noexcept
- constexpr _Tp * **data** () noexcept
- constexpr const _Tp * **end** () const noexcept
- constexpr _Tp * **end** () noexcept
- constexpr decltype(auto) **front** ()
- constexpr **operator bool** () noexcept(noexcept(ranges::empty(_M_derived())))
- constexpr decltype(auto) **operator[ ]** (range_difference_t< _Range > __n)

Static Public Member Functions

- static constexpr bool **empty** () noexcept
- static constexpr size_t **size** () noexcept

6.920.1 Detailed Description

template<copy_constructible _Tp>
requires is_object_v<_Tp>
class std::ranges::single_view< _Tp >

A view that contains exactly one element.
The documentation for this class was generated from the following file:

- ranges

6.921 std::slice Class Reference

`#include <slice_array.h>`

Public Member Functions

- slice ()
- slice (size_t __o, size_t __d, size_t __s)
- size_t size () const
- size_t start () const
- size_t stride () const

Friends

- bool operator== (const slice &, const slice &)=default

6.921.1 Detailed Description

Class defining one-dimensional subset of an array.
The slice class represents a one-dimensional subset of an array, specified by three parameters: start offset, size, and stride. The start offset is the index of the first element of the array that is part of the subset. The size is the total number of elements in the subset. Stride is the distance between each successive array element to include in the subset.
For example, with an array of size 10, and a slice with offset 1, size 3 and stride 2, the subset consists of array elements 1, 3, and 5.

6.921.2 Friends And Related Symbol Documentation

operator==

```
bool operator== (
            const slice & ,
            const slice & )  [friend]
```

Equality comparison.
References slice().
The documentation for this class was generated from the following file:

- slice_array.h

6.922 std::slice_array< _Tp > Class Template Reference

`#include <slice_array.h>`

Public Types

- typedef _Tp **value_type**

Public Member Functions

- slice_array (const slice_array &)
- template<class _Dom>
  void **operator%=** (const _Expr< _Dom, _Tp > &) const
- void operator%= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator&=** (const _Expr< _Dom, _Tp > &) const
- void operator&= (const valarray< _Tp > &) const

- template<class _Dom>
  void **operator*=** (const _Expr< _Dom, _Tp > &) const
- void operator*= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator+=** (const _Expr< _Dom, _Tp > &) const
- void operator+= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator-=** (const _Expr< _Dom, _Tp > &) const
- void operator-= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator/=** (const _Expr< _Dom, _Tp > &) const
- void operator/= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator<<=** (const _Expr< _Dom, _Tp > &) const
- void operator<<= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator=** (const _Expr< _Dom, _Tp > &) const
- void operator= (const _Tp &) const
- slice_array & operator= (const slice_array &)
- void operator= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator>>=** (const _Expr< _Dom, _Tp > &) const
- void operator>>= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator^=** (const _Expr< _Dom, _Tp > &) const
- void operator^= (const valarray< _Tp > &) const
- template<class _Dom>
  void **operator|=** (const _Expr< _Dom, _Tp > &) const
- void operator|= (const valarray< _Tp > &) const

Friends

- class **valarray< _Tp >**

6.922.1 Detailed Description

template<typename _Tp>
class std::slice_array< _Tp >

Reference to one-dimensional subset of an array.
A slice_array is a reference to the actual elements of an array specified by a slice. The way to get a slice_array is to call operator[ ](slice) on a valarray. The returned slice_array then permits carrying operations out on the referenced subset of elements in the original valarray. For example, operator+=(valarray) will add values to the subset of elements in the underlying valarray this slice_array refers to.

Parameters

Tp	Element type.

6.922.2 Member Function Documentation

operator%=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator%= (
            const valarray< _Tp > & ) const
```

Modulo slice elements by corresponding elements of *v*.

operator&=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator&= (
            const valarray< _Tp > & ) const
```

Logical and slice elements with corresponding elements of *v*.

operator*=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator*= (
            const valarray< _Tp > & ) const
```

Multiply slice elements by corresponding elements of *v*.

operator+=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator+= (
            const valarray< _Tp > & ) const
```

Add corresponding elements of *v* to slice elements.

operator-=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator-= (
            const valarray< _Tp > & ) const
```

Subtract corresponding elements of *v* from slice elements.

operator/=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator/= (
            const valarray< _Tp > & ) const
```

Divide slice elements by corresponding elements of *v*.

operator<<=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator<<= (
            const valarray< _Tp > & ) const
```

Left shift slice elements by corresponding elements of *v*.

operator>>=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator>>= (
            const valarray< _Tp > & ) const
```

Right shift slice elements by corresponding elements of *v*.

operator^=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator^= (
            const valarray< _Tp > & ) const
```

Logical xor slice elements with corresponding elements of *v*.

operator"|=()

```
template<typename _Tp>
void std::slice_array< _Tp >::operator|= (
            const valarray< _Tp > & ) const
```

Logical or slice elements with corresponding elements of *v*.
The documentation for this class was generated from the following files:

- valarray
- slice_array.h

6.923 __gnu_cxx::slist< _Tp, _Alloc > Class Template Reference

`#include <slist>`

Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef _Slist_iterator< _Tp, const _Tp &, const _Tp * > **const_iterator**
- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef _Slist_iterator< _Tp, _Tp &, _Tp * > **iterator**
- typedef value_type * **pointer**
- typedef value_type & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- template<class _InputIterator>
  slist (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- **slist** (const allocator_type &__a=allocator_type())
- **slist** (const slist &__x)
- **slist** (size_type __n)
- **slist** (size_type __n, const value_type &__x, const allocator_type &__a=allocator_type())
- template<class _InputIterator>
  void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, std::__false_type)
- template<class _Integer>
  void **_M_assign_dispatch** (_Integer __n, _Integer __val, std::__true_type)
- void **_M_fill_assign** (size_type __n, const _Tp &__val)
- template<class _InputIterator>
  void **assign** (_InputIterator __first, _InputIterator __last)
- void **assign** (size_type __n, const _Tp &__val)
- iterator **before_begin** ()

- const_iterator **before_begin** () const
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator __first, iterator __last)
- iterator **erase** (iterator __pos)
- iterator **erase_after** (iterator __before_first, iterator __last)
- iterator **erase_after** (iterator __pos)
- reference **front** ()
- const_reference **front** () const
- allocator_type **get_allocator** () const
- iterator **insert** (iterator __pos)
- template<class _InIterator>
  void **insert** (iterator __pos, _InIterator __first, _InIterator __last)
- iterator **insert** (iterator __pos, const value_type &__x)
- void **insert** (iterator __pos, size_type __n, const value_type &__x)
- iterator **insert_after** (iterator __pos)
- template<class _InIterator>
  void **insert_after** (iterator __pos, _InIterator __first, _InIterator __last)
- iterator **insert_after** (iterator __pos, const value_type &__x)
- void **insert_after** (iterator __pos, size_type __n, const value_type &__x)
- size_type **max_size** () const
- template<class _StrictWeakOrdering>
  void **merge** (slist &, _StrictWeakOrdering)
- void **merge** (slist &__x)
- slist & **operator=** (const slist &__x)
- void **pop_front** ()
- iterator **previous** (const_iterator __pos)
- const_iterator **previous** (const_iterator __pos) const
- void **push_front** ()
- void **push_front** (const value_type &__x)
- void **remove** (const _Tp &__val)
- template<class _Predicate>
  void **remove_if** (_Predicate __pred)
- void **resize** (size_type new_size)
- void **resize** (size_type new_size, const _Tp &__x)
- void **reverse** ()
- size_type **size** () const
- void **sort** ()
- template<class _StrictWeakOrdering>
  void **sort** (_StrictWeakOrdering __comp)
- void **splice** (iterator __pos, slist &__x)
- void **splice** (iterator __pos, slist &__x, iterator __first, iterator __last)
- void **splice** (iterator __pos, slist &__x, iterator __i)
- void **splice_after** (iterator __pos, iterator __before_first, iterator __before_last)
- void **splice_after** (iterator __pos, iterator __prev)
- void **splice_after** (iterator __pos, slist &__x)
- void **swap** (slist &__x)
- void **unique** ()
- template<class _BinaryPredicate>
  void **unique** (_BinaryPredicate __pred)

Static Private Member Functions

- static constexpr size_type max_size (const _Alloc &__a) noexcept

6.923.1 Detailed Description

template<class _Tp, class _Alloc = std::allocator<_Tp>>
class __gnu_cxx::slist< _Tp, _Alloc >

This is an SGI extension.

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

The documentation for this class was generated from the following file:

- slist

6.924 std::experimental::filesystem::v1::space_info Struct Reference

`#include <fs_fwd.h>`

Public Attributes

- uintmax_t **available**
- uintmax_t **capacity**
- uintmax_t **free**

6.924.1 Detailed Description

Information about free space on a disk.
The documentation for this struct was generated from the following file:

- experimental/bits/fs_fwd.h

6.925 std::filesystem::space_info Struct Reference

`#include <fs_fwd.h>`

Public Attributes

- uintmax_t **available**
- uintmax_t **capacity**
- uintmax_t **free**

Friends

- bool **operator==** (const space_info &, const space_info &)=default

6.925.1 Detailed Description

Information about free space on a disk.
The documentation for this struct was generated from the following file:

- bits/fs_fwd.h

6.926 __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > Class Template Reference

`#include <splay_tree_.hpp>`

Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef std::pair< size_type, size_type > **comp_hash**
- typedef base_type::const_iterator **const_iterator**
- typedef base_type::const_pointer **const_pointer**
- typedef base_type::const_reference **const_reference**
- typedef base_type::const_reverse_iterator **const_reverse_iterator**
- typedef splay_tree_tag **container_category**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::key_const_pointer **key_const_pointer**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_pointer **key_pointer**
- typedef base_type::key_reference **key_reference**
- typedef base_type::key_type **key_type**
- typedef base_type::mapped_const_pointer **mapped_const_pointer**
- typedef base_type::mapped_const_reference **mapped_const_reference**
- typedef base_type::mapped_pointer **mapped_pointer**
- typedef base_type::mapped_reference **mapped_reference**
- typedef base_type::mapped_type **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef traits_type::node_const_iterator **node_const_iterator**
- typedef traits_type::node_iterator **node_iterator**
- typedef base_type::node_update **node_update**
- typedef base_type::const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef base_type::pointer **pointer**
- typedef base_type::reference **reference**
- typedef base_type::reverse_iterator **reverse_iterator**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef base_type::value_type **value_type**

Public Member Functions

- **splay_tree_map** (const Cmp_Fn &)
- **splay_tree_map** (const Cmp_Fn &, const node_update &)
- **splay_tree_map** (const splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- iterator **begin** ()
- const_iterator **begin** () const
- void **clear** ()
- template<typename It>
  void **copy_from_range** (It, It)
- bool **empty** () const

- iterator **end** ()
- const_iterator **end** () const
- iterator **erase** (iterator it)
- bool **erase** (key_const_reference)
- reverse_iterator **erase** (reverse_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- point_iterator **find** (key_const_reference)
- point_const_iterator **find** (key_const_reference) const
- Cmp_Fn & **get_cmp_fn** ()
- const Cmp_Fn & **get_cmp_fn** () const
- void **initialize** ()
- std::pair< point_iterator, bool > **insert** (const_reference r_value)
- void **join** (splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **lower_bound** (key_const_reference)
- point_const_iterator **lower_bound** (key_const_reference) const
- size_type **max_size** () const
- node_iterator node_begin ()
- node_const_iterator node_begin () const
- node_iterator node_end ()
- node_const_iterator node_end () const
- mapped_reference **operator[ ]** (key_const_reference r_key)
- reverse_iterator **rbegin** ()
- const_reverse_iterator **rbegin** () const
- reverse_iterator **rend** ()
- const_reverse_iterator **rend** () const
- size_type **size** () const
- void **split** (key_const_reference, splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- void **swap** (splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- point_iterator **upper_bound** (key_const_reference)
- point_const_iterator **upper_bound** (key_const_reference) const

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

Protected Types

- typedef node_alloc_traits::value_type **node**
- typedef node_alloc_traits::allocator_type **node_allocator**
- typedef traits_type::null_node_update_pointer **null_node_update_pointer**
- typedef types_traits< Key, Mapped, _Alloc, false > **traits_base**

Protected Member Functions

- void **actual_erase_node** (node_pointer)
- template<typename Node_Update_>
  void **apply_update** (node_pointer, Node_Update_ *)
- void **apply_update** (node_pointer, null_node_update_pointer)
- std::pair< node_pointer, bool > **erase** (node_pointer)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, false_type)
- node_pointer **get_new_node_for_leaf_insert** (const_reference, true_type)
- void **initialize_min_max** ()
- iterator **insert_imp_empty** (const_reference)
- std::pair< point_iterator, bool > **insert_leaf** (const_reference)
- iterator **insert_leaf_new** (const_reference, node_pointer, bool)
- void **join_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **join_prep** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- size_type **recursive_count** (node_pointer) const
- void **rotate_left** (node_pointer)
- void **rotate_parent** (node_pointer)
- void **rotate_right** (node_pointer)
- void **split_finish** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)
- bool **split_prep** (key_const_reference, bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _↩ Alloc > &)
- void **update_min_max_for_erased_node** (node_pointer)
- void **update_subtree_size** (node_pointer)
- template<typename Node_Update_>
  void **update_to_top** (node_pointer, Node_Update_ *)
- void **update_to_top** (node_pointer, null_node_update_pointer)
- void **value_swap** (bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc > &)

Static Protected Member Functions

- static void **clear_imp** (node_pointer)

Protected Attributes

- node_pointer **m_p_head**
- size_type **m_size**

Static Protected Attributes

- static node_allocator **s_node_allocator**

6.926.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩ Alloc>
class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Splay tree.

6.926.2 Member Function Documentation

node_begin() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↩
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_begin
() [inline], [inherited]
```

Returns a node_iterator corresponding to the node at the root of the tree.

node_begin() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↩
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↩
begin () const [inline], [inherited]
```

Returns a const node_iterator corresponding to the node at the root of the tree.

node_end() [1/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_iterator __gnu_↩
pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_end ()
[inline], [inherited]
```

Returns a node_iterator corresponding to a node just after a leaf of the tree.

node_end() [2/2]

```
template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _↩
Alloc>
bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_const_iterator __↩
gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_↩
end () const [inline], [inherited]
```

Returns a const node_iterator corresponding to a node just after a leaf of the tree.

The documentation for this class was generated from the following file:

- splay_tree_.hpp

6.927 __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc > Struct Template Reference

```
#include <node.hpp>
```

Public Types

- typedef rebind_traits< _Alloc, metadata_type >::const_reference **metadata_const_reference**
- typedef rebind_traits< _Alloc, metadata_type >::reference **metadata_reference**
- typedef Metadata **metadata_type**
- typedef rebind_traits< _Alloc, splay_tree_node_ >::pointer **node_pointer**
- typedef rebind_traits< _Alloc, splay_tree_node_ >::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- metadata_reference **get_metadata** ()
- metadata_const_reference **get_metadata** () const
- bool **special** () const

Public Attributes

- metadata_type **m_metadata**
- node_pointer **m_p_left**
- node_pointer **m_p_parent**
- node_pointer **m_p_right**
- bool **m_special**
- size_type **m_subtree_size**
- value_type **m_value**

6.927.1 Detailed Description

template<typename Value_Type, class Metadata, typename _Alloc>
struct __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >

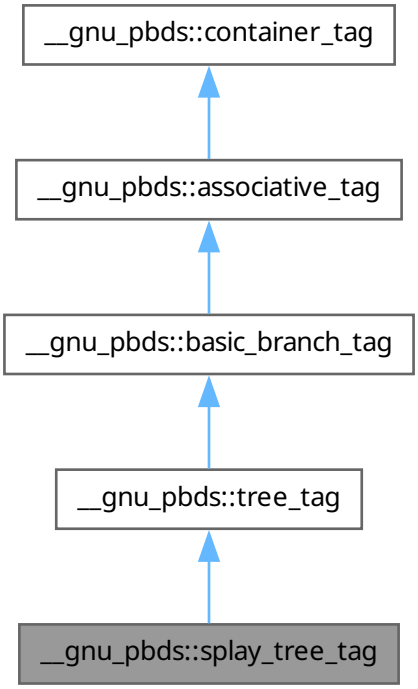
Node for splay tree.
The documentation for this struct was generated from the following file:

- splay_tree_/node.hpp

6.928 __gnu_pbds::splay_tree_tag Struct Reference

`#include <tag_and_trait.hpp>`

Inheritance diagram for __gnu_pbds::splay_tree_tag:



6.928.1 Detailed Description

Splay tree.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.929 std::stack< _Tp, _Sequence > Class Template Reference

`#include <stl_stack.h>`

Public Types

- typedef _Sequence::const_reference **const_reference**
- typedef _Sequence **container_type**
- typedef _Sequence::reference **reference**
- typedef _Sequence::size_type **size_type**
- typedef _Sequence::value_type **value_type**

Public Member Functions

- template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<_Seq>::value>::type> stack ()

- template<typename _InputIterator, typename _Alloc, typename = _RequireInputIter<_InputIterator>, typename = _Uses<_Alloc>>
  stack (_InputIterator __first, _InputIterator __last, const _Alloc &__a)
- **stack** (_Sequence &&__c)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  stack (_Sequence &&__c, const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  stack (const _Alloc &__a)
- **stack** (const _Sequence &__c)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  stack (const _Sequence &__c, const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  stack (const stack &__q, const _Alloc &__a)
- template<typename _Alloc, typename _Requires = _Uses<_Alloc>>
  stack (stack &&__q, const _Alloc &__a)
- template<typename... _Args>
  decltype(auto) **emplace** (_Args &&... __args)
- bool empty () const
- void pop ()
- void push (const value_type &__x)
- void **push** (value_type &&__x)
- size_type size () const
- void **swap** (stack &__s) noexcept(__is_nothrow_swappable< _Sequence >::value)
- reference top ()
- const_reference top () const

Protected Attributes

- _Sequence **c**

Friends

- template<typename _Tp1, typename _Seq1>
  bool **operator**< (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)
- template<typename _Tp1, three_way_comparable _Seq1>
  compare_three_way_result_t< _Seq1 > **operator**<=> (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)
- template<typename _Tp1, typename _Seq1>
  bool **operator==** (const stack< _Tp1, _Seq1 > &, const stack< _Tp1, _Seq1 > &)

6.929.1 Detailed Description

template<typename _Tp, typename _Sequence = deque<_Tp>>
class std::stack< _Tp, _Sequence >

A standard container giving FILO behavior.

Template Parameters

_Tp	Type of element.
_Sequence	Type of underlying sequence, defaults to deque<_Tp>.

Meets many of the requirements of a container, but does not define anything to do with iterators. Very few of the other standard container interfaces are defined.

This is not a true container, but an *adaptor*. It holds another container, and provides a wrapper interface to that container. The wrapper is what enforces strict first-in-last-out stack behavior.
The second template parameter defines the type of the underlying sequence/container. It defaults to std::deque, but it can be any type that supports `back`, `push_back`, and `pop_back`, such as std::list, std::vector, or an appropriate user-defined type.
Members not found in *normal* containers are `container_type`, which is a typedef for the second Sequence parameter, and `push`, `pop`, and `top`, which are standard stack/FILO operations.

6.929.2 Constructor & Destructor Documentation

stack()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
template<typename _Seq = _Sequence, typename _Requires = typename enable_if<is_default_constructible<↩
_Seq>::value>::type>
std::stack< _Tp, _Sequence >::stack ()  [inline]
```

Default constructor creates no elements.

6.929.3 Member Function Documentation

empty()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
bool std::stack< _Tp, _Sequence >::empty () const  [inline], [nodiscard]
```

Returns true if the stack is empty.

pop()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack< _Tp, _Sequence >::pop ()  [inline]
```

Removes first element.
This is a typical stack operation. It shrinks the stack by one. The time complexity of the operation depends on the underlying sequence.
Note that no data is returned, and if the first element's data is needed, it should be retrieved before pop() is called.

push()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
void std::stack< _Tp, _Sequence >::push (
            const value_type & __x)  [inline]
```

Add data to the top of the stack.

Parameters

_↩ _x	Data to be added.

This is a typical stack operation. The function creates an element at the top of the stack and assigns the given data to it. The time complexity of the operation depends on the underlying sequence.

size()

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
size_type std::stack< _Tp, _Sequence >::size () const  [inline], [nodiscard]
```

Returns the number of elements in the stack.

top() [1/2]

```
template<typename _Tp, typename _Sequence = deque<_Tp>>
reference std::stack< _Tp, _Sequence >::top () [inline], [nodiscard]
```

Returns a read/write reference to the data at the first element of the stack.

top() [2/2]

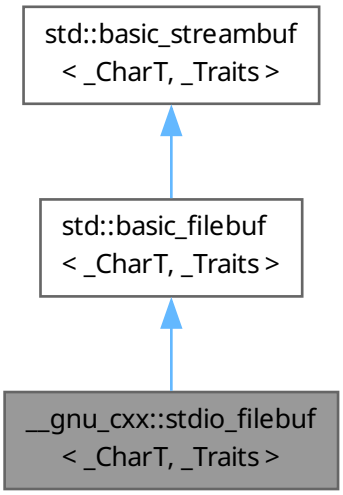
```
template<typename _Tp, typename _Sequence = deque<_Tp>>
const_reference std::stack< _Tp, _Sequence >::top () const [inline], [nodiscard]
```

Returns a read-only (constant) reference to the data at the first element of the stack.
The documentation for this class was generated from the following file:

- stl_stack.h

6.930 __gnu_cxx::stdio_filebuf< _CharT, _Traits > Class Template Reference

`#include <stdio_filebuf.h>`
Inheritance diagram for __gnu_cxx::stdio_filebuf< _CharT, _Traits >:



Public Types

- typedef codecvt< char_type, char, __state_type > **__codecvt_type**
- typedef __basic_file< char > **__file_type**
- typedef basic_filebuf< char_type, traits_type > **__filebuf_type**
- typedef traits_type::state_type **__state_type**
- typedef basic_streambuf< char_type, traits_type > **__streambuf_type**
- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**

- typedef std::size_t **size_t**
- typedef _Traits **traits_type**

Public Member Functions

- stdio_filebuf ()
- stdio_filebuf (int __fd, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))
- stdio_filebuf (std::__c_file *__f, std::ios_base::openmode __mode, size_t __size=static_cast< size_t >(BUFSIZ))
- **stdio_filebuf** (stdio_filebuf &&)=default
- virtual ~stdio_filebuf ()
- __filebuf_type * close ()
- int fd ()
- std::__c_file * file ()
- locale getloc () const
- streamsize in_avail ()
- bool is_open () const throw ()
- template<typename _Path>
  _If_fs_path< _Path, __filebuf_type * > open (const _Path &__s, ios_base::openmode __mode)
- __filebuf_type * open (const char *__s, ios_base::openmode __mode)
- __filebuf_type * open (const std::string &__s, ios_base::openmode __mode)
- stdio_filebuf & **operator=** (stdio_filebuf &&)=default
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)
- int_type sungetc ()
- void **swap** (stdio_filebuf &__fb)
- void **swap** (basic_filebuf &)

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void **_M_allocate_internal_buffer** ()
- bool **_M_convert_to_external** (char_type *, streamsize)
- void _M_create_pback ()
- void **_M_destroy_internal_buffer** () throw ()
- void _M_destroy_pback () throw ()
- int **_M_get_ext_pos** (__state_type &__state)
- pos_type **_M_seek** (off_type __off, ios_base::seekdir __way, __state_type __state)
- void _M_set_buffer (streamsize __off)

- bool **_M_terminate_output** ()
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=_Traits::eof())
- virtual int_type pbackfail (int_type __c=_Traits::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::↩ in|ios_base::out)
- virtual pos_type seekpos (pos_type __pos, ios_base::openmode __mode=ios_base::in|ios_base::out)
- virtual __streambuf_type * setbuf (char_type *__s, streamsize __n)
- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- virtual int sync ()
- virtual int_type uflow ()
- virtual int_type underflow ()
- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)

- char_type * eback () const
- char_type * gptr () const
- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- char_type * _M_buf
- bool **_M_buf_allocated**
- locale _M_buf_locale
- size_t _M_buf_size
- const __codecvt_type * **_M_codecvt**
- char * _M_ext_buf
- streamsize _M_ext_buf_size
- char * **_M_ext_end**
- const char * _M_ext_next
- __file_type **_M_file**
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- __c_lock **_M_lock**
- ios_base::openmode _M_mode
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end
- bool _M_reading

- __state_type **_M_state_beg**
- __state_type **_M_state_cur**
- __state_type **_M_state_last**
- bool **_M_writing**

- char_type _M_pback
- char_type * _M_pback_cur_save
- char_type * _M_pback_end_save
- bool _M_pback_init

6.930.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_filebuf< _CharT, _Traits >

Provides a layer of compatibility for C/POSIX.
This GNU extension provides extensions for working with standard C FILE*'s and POSIX file descriptors. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.

6.930.2 Constructor & Destructor Documentation

stdio_filebuf() [1/3]

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf ()  [inline]
```

deferred initialization
References std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

stdio_filebuf() [2/3]

```
template<typename _CharT, typename _Traits>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (
            int __fd,
            std::ios_base::openmode __mode,
            size_t __size = static_cast<size_t>( BUFSIZ ))
```

Parameters

__fd	An open file descriptor.
__mode	Same meaning as in a standard filebuf.
__size	Optimal or preferred size of internal buffer, in chars.

This constructor associates a file stream buffer with an open POSIX file descriptor. The file descriptor will be automatically closed when the stdio_filebuf is closed/destroyed.
References std::basic_filebuf< _CharT, _Traits >::_M_buf_size, std::basic_filebuf< _CharT, _Traits >::_M_mode, std::basic_filebuf< _CharT, _Traits >::_M_reading, std::basic_filebuf< _CharT, _Traits >::_M_set_buffer(), and std::basic_filebuf< _CharT, _Traits >::is_open().

stdio_filebuf() [3/3]

```
template<typename _CharT, typename _Traits>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf (
            std::__c_file * __f,
```

```
            std::ios_base::openmode __mode,
            size_t __size = static_cast<size_t>( BUFSIZ ))
```

Parameters

__f	An open FILE*.
__mode	Same meaning as in a standard filebuf.
__size	Optimal or preferred size of internal buffer, in chars. Defaults to system's BUFSIZ.

This constructor associates a file stream buffer with an open C FILE*. The FILE* will not be automatically closed when the stdio_filebuf is closed/destroyed.
References std::basic_filebuf< _CharT, _Traits >::_M_buf_size, std::basic_filebuf< _CharT, _Traits >::_M_mode, std::basic_filebuf< _CharT, _Traits >::_M_reading, std::basic_filebuf< _CharT, _Traits >::_M_set_buffer(), and std::basic_filebuf< _CharT, _Traits >::is_open().

~stdio_filebuf()

```
template<typename _CharT, typename _Traits>
__gnu_cxx::stdio_filebuf< _CharT, _Traits >::~stdio_filebuf ()  [virtual]
```
Closes the external data stream if the file descriptor constructor was used.

6.930.3 Member Function Documentation

_M_create_pback()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_create_pback ()  [inline], [protected], [inherited]
```
Initializes pback buffers, and moves normal buffers to safety. Assumptions: _M_in_cur has already been moved back
Referenced by pbackfail().

_M_destroy_pback()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_destroy_pback () throw ( )   [inline], [protected],
[inherited]
```
Deactivates pback buffer contents, and restores normal buffer. Assumptions: The pback buffer has only moved forward.
Referenced by overflow(), seekoff(), seekpos(), underflow(), and xsgetn().

_M_set_buffer()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::_M_set_buffer (
            streamsize __off)  [inline], [protected], [inherited]
```
This function sets the pointers of the internal buffer, both get and put areas. Typically:
__off == egptr() - eback() upon underflow/uflow (**read** mode); __off == 0 upon overflow (**write** mode); __off == -1 upon open, setbuf, seekoff/pos (**uncommitted** mode).
NB: epptr() - pbase() == _M_buf_size - 1, since _M_buf_size reflects the actual allocated memory and the last cell is reserved for the overflow char of a full put area.
Referenced by __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), close(), imbue(), open(), overflow(), pbackfail(), underflow(), xsgetn(), and xsputn().

close()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::close
() [inherited]
```

Closes the currently associated file.

Returns

`this` on success, NULL on failure

If no file is currently open, this function immediately fails.
If a *put buffer area* exists, `overflow(eof)` is called to flush all the characters. The file is then closed.
If any operations fail, this function also fails.
References basic_filebuf(), _M_mode, _M_pback_init, _M_reading, _M_set_buffer(), and is_open().
Referenced by std::basic_filebuf< char_type, traits_type >::~basic_filebuf(), and open().

eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _Cha std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_streambuf< char_type, traits_type >::in_avail std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< char std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sputc(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

fd()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
int __gnu_cxx::stdio_filebuf< _CharT, _Traits >::fd ()  [inline]
```

Returns

The underlying file descriptor.

Once associated with an external data stream, this function can be used to access the underlying POSIX file descriptor. Note that there is no way for the library to track what you do with the descriptor, so be careful.

file()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
std::__c_file * __gnu_cxx::stdio_filebuf< _CharT, _Traits >::file ()  [inline]
```

Returns

The underlying FILE*.

This function can be used to access the underlying "C" file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_streambuf< char_type, traits_type >::uflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const  [inline], [inherited]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.
Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy
std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_stringbuf< _CharT,
std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< cha
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_streambuf< char_type, traits_type >::sgetc(),
std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< char >::showmanyc(), std::basic_streambuf< char_type, traits
std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_streambuf< char_type, traits_type >::uflow(),
std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(),
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and
xsgetn().

imbue()

```
template<typename _CharT, typename _Traits>
void std::basic_filebuf< _CharT, _Traits >::imbue (
             const locale & __loc)  [protected], [virtual], [inherited]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_ext_buf, _M_ext_next, _M_mode, _M_reading, _M_set_buffer(), std::ios_base::cur, std::basic_streambuf< _CharT, _Trait
std::basic_streambuf< _CharT, _Traits >::gptr(), is_open(), and seekoff().

in_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ()  [inline], [inherited]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

is_open()

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::is_open () const throw ( )   [inline], [nodiscard],
[inherited]
```

Returns true if the external file is open.

Referenced by __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), close(), imbue(), open(), seekoff(), seekpos(), setbuf(), and showmanyc().

open() [1/3]

```
template<typename _CharT, typename _Traits>
template<typename _Path>
_If_fs_path< _Path, __filebuf_type * > std::basic_filebuf< _CharT, _Traits >::open (
            const _Path & __s,
            ios_base::openmode __mode)  [inline], [inherited]
```

Opens an external file.

Parameters

__s	The name of the file, as a filesystem::path.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

open() [2/3]

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
            const char * __s,
            ios_base::openmode __mode)  [inherited]
```

Opens an external file.

Parameters

__s	The name of the file.

__mode	The open mode flags.

Returns

`this` on success, NULL on failure

If a file is already open, this function immediately fails. Otherwise it tries to open the file named *__s* using the flags given in *__mode*.
Table 92, adapted here, gives the relation between openmode combinations and the equivalent `fopen()` flags. (NB: lines app, in|out|app, in|app, binary|app, binary|in|out|app, and binary|in|app per DR 596)

```
+-------------------------------------------------------+
| ios_base Flag combination            stdio equivalent |
|binary  in  out  trunc  app                            |
+-------------------------------------------------------+
|             +                        w                |
|             +           +            a                |
|                         +            a                |
|             +     +                  w                |
|          +                           r                |
|          +   +                       r+               |
|          +   +     +                 w+               |
|          +   +           +           a+               |
|          +               +           a+               |
+-------------------------------------------------------+
|    +        +                        wb               |
|    +        +           +            ab               |
|    +                    +            ab               |
|    +        +     +                  wb               |
|    +     +                           rb               |
|    +     +   +                       r+b              |
|    +     +   +     +                 w+b              |
|    +     +   +           +           a+b              |
|    +     +               +           a+b              |
+-------------------------------------------------------+
```

References _M_mode, _M_reading, _M_set_buffer(), std::ios_base::ate, close(), std::ios_base::end, is_open(), and seekoff().
Referenced by std::basic_filebuf< char_type, traits_type >::open(), and std::basic_filebuf< char_type, traits_type >::open().

open() [3/3]

```
template<typename _CharT, typename _Traits>
__filebuf_type * std::basic_filebuf< _CharT, _Traits >::open (
            const std::string & __s,
            ios_base::openmode __mode)  [inline], [inherited]
```

Opens an external file.

Parameters

__s	The name of the file.
__mode	The open mode flags.

Returns

`this` on success, NULL on failure

overflow()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::overflow (
            int_type __c = _Traits::eof())  [protected], [virtual], [inherited]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

_↩ _c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually __c, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if __c is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_destroy_pback(), _M_mode, _M_reading, _M_set_buffer(), std::ios_base::app, std::ios_base::cur, std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase(), std::basic_streambuf< _CharT, _Traits >::pbum and std::basic_streambuf< _CharT, _Traits >::pptr().
Referenced by pbackfail(), sync(), underflow(), and xsgetn().

pbackfail()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::pbackfail (
            int_type __c = _Traits::eof())  [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

Parameters

_↩ _c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_create_pback(), _M_mode, _M_pback_init, _M_reading, _M_set_buffer(), std::ios_base::cur, std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::gbump(), std::basic_streambuf< _CharT, _
std::ios_base::in, overflow(), seekoff(), and underflow().

pbase()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, std::basic_stringbuf< char >::str(), std::basic_filebuf< _CharT, _Traits >::sync(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), and std::basic_streambuf< char_type, traits_type >::sputc().

pptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_streambuf< char_t std::basic_filebuf< _CharT, _Traits >::sync(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
            const locale & __loc)  [inline], [inherited]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
```

```
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ()  [inline], [inherited]
```

Calls virtual sync function.
Referenced by std::basic_istream< _CharT, _Traits >::sync().

sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_streambuf< char_type, traits_type >::snextc().

seekoff()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekoff (
            off_type ,
            ios_base::seekdir ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [protected], [virtual], [inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_destroy_pback(), _M_reading, std::ios_base::cur, is_open(), std::basic_streambuf< _CharT, _Traits >::pbase(), and std::basic_streambuf< _CharT, _Traits >::pptr().
Referenced by imbue(), open(), and pbackfail().

seekpos()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::pos_type std::basic_filebuf< _CharT, _Traits >::seekpos (
            pos_type ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [protected], [virtual], [inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_destroy_pback(), std::ios_base::beg, and is_open().

setbuf()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::__streambuf_type * std::basic_filebuf< _CharT, _Traits >::↩
setbuf (
            char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Manipulates the buffer.

Parameters

_↩ _s	Pointer to a buffer area.
_↩ _n	Size of __s.

Returns

`this`

If no file has been opened, and both __s and __n are zero, then the stream becomes unbuffered. Otherwise, __s is used as a buffer; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html↩ #io.streambuf.buffering for more.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf, _M_buf_size, and is_open().

setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
            char_type * __gbeg,
            char_type * __gnext,
            char_type * __gend)  [inline], [protected], [inherited]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), std::basic_filebuf< char_type, traits_type >::_M_create
std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_filebuf< char_type, traits_type >::_M_set_buffer(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
            char_type * __pbeg,
            char_type * __pend)  [inline], [protected], [inherited]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), and std::basic_filebuf< char_type, traits_type >::_M_s

sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::get(),
std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::ig
and std::basic_streambuf< char_type, traits_type >::snextc().

sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for xsgetn.

Parameters

_↩ _s	A buffer area.
_↩ _n	A count.

Returns xsgetn(__s,__n). The effect is to fill __*s*[0] through __*s*[__n-1] with characters from the input sequence, if possible.

showmanyc()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::showmanyc ()  [protected], [virtual], [inherited]
```

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_mode, std::ios_base::binary, std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits > std::ios_base::in, and is_open().

snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, char_traits< _CharT > >::get(), and std::basic_istream< _CharT, char_traits< _CharT > >::readsome().

sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
            char_type __c)  [inline], [inherited]
```

Pushing characters back into the input stream.

Parameters

_↩ _c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but __c is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be __c.
Referenced by std::basic_istream< _CharT, _Traits >::putback().

sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
            char_type __c)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _c	A character to output.

Returns

__c, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.
Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow().

sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
            const char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _s	A buffer read area.
_↩ _n	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __s[0] through __s[__n-1] to the output sequence, if possible.

sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ()  [inline], [inherited]
```
Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.
Referenced by std::basic_istream< _CharT, _Traits >::unget().

sync()

```
template<typename _CharT, typename _Traits>
int std::basic_filebuf< _CharT, _Traits >::sync ()  [protected], [virtual], [inherited]
```
Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References overflow(), std::basic_streambuf< _CharT, _Traits >::pbase(), and std::basic_streambuf< _CharT, _Traits >::pptr().

uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ()  [inline], [protected], [virtual],
[inherited]
```
Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.
Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.
Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc(), and xsgetn().

underflow()

```
template<typename _CharT, typename _Traits>
basic_filebuf< _CharT, _Traits >::int_type std::basic_filebuf< _CharT, _Traits >::underflow ()
[protected], [virtual], [inherited]
```
Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_destroy_pback(), _M_mode, _M_set_buffer(), std::basic_streambuf< _CharT, _Traits >::eback(), std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::gptr(), std::ios_base::in, and overflow().
Referenced by pbackfail().

xsgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsgetn (
            char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character extraction.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __s[0] through __s[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __n characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_destroy_pback(), _M_mode, _M_pback_init, _M_reading, _M_set_buffer(), std::basic_streambuf< _CharT, _
std::basic_streambuf< _CharT, _Traits >::egptr(), std::basic_streambuf< _CharT, _Traits >::gbump(), std::basic_streambuf< _CharT, _T
std::ios_base::in, overflow(), std::basic_streambuf< _CharT, _Traits >::setg(), and std::basic_streambuf< char_type, traits_type >::xsgetn

xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character insertion.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __s[0] through __s[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References _M_buf_size, _M_mode, _M_reading, _M_set_buffer(), std::ios_base::app, std::basic_streambuf< _CharT, _Traits >::epptr(), std::ios_base::out, std::basic_streambuf< _CharT, _Traits >::pbase(), std::basic_streambuf< _CharT, _Traits >::pptr(), and std::basic_streambuf< char_type, traits_type >::xsputn().

6.930.4 Member Data Documentation

_M_buf

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_buf  [protected], [inherited]
```

Pointer to the beginning of internal buffer.
Referenced by basic_filebuf(), and setbuf().

_M_buf_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale  [protected], [inherited]
```

Current locale setting.
Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

_M_buf_size

```
template<typename _CharT, typename _Traits>
size_t std::basic_filebuf< _CharT, _Traits >::_M_buf_size  [protected], [inherited]
```

Actual size of internal buffer. This number is equal to the size of the put area + 1 position, reserved for the overflow char of a full area.
Referenced by basic_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::
overflow(), setbuf(), underflow(), xsgetn(), and xsputn().

_M_ext_buf

```
template<typename _CharT, typename _Traits>
char* std::basic_filebuf< _CharT, _Traits >::_M_ext_buf  [protected], [inherited]
```

Buffer for external characters. Used for input when codecvt::always_noconv() == false. When valid, this corresponds to eback().
Referenced by basic_filebuf(), and imbue().

_M_ext_buf_size

```
template<typename _CharT, typename _Traits>
streamsize std::basic_filebuf< _CharT, _Traits >::_M_ext_buf_size  [protected], [inherited]
```

Size of buffer held by _M_ext_buf.
Referenced by basic_filebuf().

_M_ext_next

```
template<typename _CharT, typename _Traits>
const char* std::basic_filebuf< _CharT, _Traits >::_M_ext_next  [protected], [inherited]
```

Pointers into the buffer held by _M_ext_buf that delimit a subsequence of bytes that have been read but not yet converted. When valid, _M_ext_next corresponds to egptr().
Referenced by basic_filebuf(), and imbue().

_M_in_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg  [protected], [inherited]
```

Start of get area.

_M_in_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur  [protected], [inherited]
```

Current read area.

_M_in_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end  [protected], [inherited]
```

End of get area.

_M_mode

```
template<typename _CharT, typename _Traits>
ios_base::openmode std::basic_filebuf< _CharT, _Traits >::_M_mode  [protected], [inherited]
```

Place to stash in || out || in | out settings for current filebuf.
Referenced by basic_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::
close(), imbue(), open(), overflow(), pbackfail(), showmanyc(), underflow(), xsgetn(), and xsputn().

_M_out_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg  [protected], [inherited]
```

Start of put area.

_M_out_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur  [protected], [inherited]
```

Current put area.

_M_out_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end  [protected], [inherited]
```

End of put area.

_M_pback

```
template<typename _CharT, typename _Traits>
char_type std::basic_filebuf< _CharT, _Traits >::_M_pback  [protected], [inherited]
```

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf().

_M_pback_cur_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_cur_save  [protected], [inherited]
```

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf().

_M_pback_end_save

```
template<typename _CharT, typename _Traits>
char_type* std::basic_filebuf< _CharT, _Traits >::_M_pback_end_save  [protected], [inherited]
```

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf().

_M_pback_init

```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_pback_init  [protected], [inherited]
```

Necessary bits for putback buffer management.

Note

pbacks of over one character are not currently supported.

Referenced by basic_filebuf(), close(), pbackfail(), and xsgetn().

_M_reading

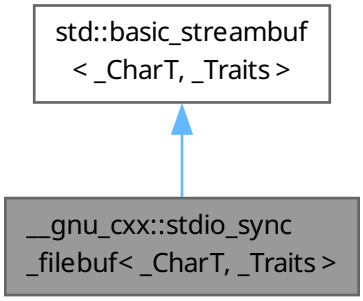
```
template<typename _CharT, typename _Traits>
bool std::basic_filebuf< _CharT, _Traits >::_M_reading  [protected], [inherited]
```

_M_reading == false && _M_writing == false for **uncommitted** mode; _M_reading == true for **read** mode; _M_writing == true for **write** mode;
NB: _M_reading == true && _M_writing == true is unused.
Referenced by basic_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::stdio_filebuf(), __gnu_cxx::stdio_filebuf< _CharT, _Traits >::
close(), imbue(), open(), overflow(), pbackfail(), seekoff(), xsgetn(), and xsputn().
The documentation for this class was generated from the following file:

- stdio_filebuf.h

6.931 __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits > Class Template Reference

`#include <stdio_sync_filebuf.h>`
Inheritance diagram for __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >:



Public Types

- typedef _CharT **char_type**
- typedef traits_type::int_type **int_type**
- typedef traits_type::off_type **off_type**
- typedef traits_type::pos_type **pos_type**
- typedef _Traits **traits_type**

Public Member Functions

- **stdio_sync_filebuf** (std::__c_file *__f)
- **stdio_sync_filebuf** (stdio_sync_filebuf &&__fb) noexcept
- std::__c_file * file ()
- locale getloc () const
- streamsize in_avail ()
- stdio_sync_filebuf & **operator=** (stdio_sync_filebuf &&__fb) noexcept
- locale pubimbue (const locale &__loc)
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)
- int_type sungetc ()
- void **swap** (stdio_sync_filebuf &__fb)

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- virtual int_type overflow (int_type __c=traits_type::eof())
- virtual int_type pbackfail (int_type __c=traits_type::eof())
- void pbump (int __n)
- virtual std::streampos **seekoff** (std::streamoff __off, std::ios_base::seekdir __dir, std::ios_base::openmode=std::ios_base::in|std::ios
- virtual pos_type seekoff (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)
- virtual std::streampos **seekpos** (std::streampos __pos, std::ios_base::openmode __mode=std::ios_base::in|std::ios_base::out)
- virtual pos_type seekpos (pos_type, ios_base::openmode=ios_base::in|ios_base::out)
- virtual basic_streambuf< char_type, _Traits > * setbuf (char_type *, streamsize)
- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- virtual int sync ()
- stdio_sync_filebuf< wchar_t >::int_type **syncgetc** ()
- stdio_sync_filebuf< char >::int_type **syncgetc** ()
- int_type **syncgetc** ()
- stdio_sync_filebuf< wchar_t >::int_type **syncputc** (int_type __c)
- stdio_sync_filebuf< char >::int_type **syncputc** (int_type __c)
- int_type **syncputc** (int_type __c)
- stdio_sync_filebuf< wchar_t >::int_type **syncungetc** (int_type __c)
- stdio_sync_filebuf< char >::int_type **syncungetc** (int_type __c)
- int_type **syncungetc** (int_type __c)
- virtual int_type uflow ()
- virtual int_type underflow ()
- std::streamsize **xsgetn** (char *__s, std::streamsize __n)
- virtual std::streamsize **xsgetn** (char_type *__s, std::streamsize __n)
- std::streamsize **xsgetn** (wchar_t *__s, std::streamsize __n)
- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- std::streamsize **xsputn** (const char *__s, std::streamsize __n)
- virtual std::streamsize **xsputn** (const char_type *__s, std::streamsize __n)
- std::streamsize **xsputn** (const wchar_t *__s, std::streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)

- char_type * eback () const
- char_type * gptr () const
- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- locale _M_buf_locale
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end

6.931.1 Detailed Description

template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
class __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >

Provides a layer of compatibility for C.
This GNU extension provides extensions for working with standard C FILE*'s. It must be instantiated by the user with the type of character used in the file stream, e.g., stdio_filebuf<char>.

6.931.2 Member Function Documentation

eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _Cha std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_streambuf< char_type, traits_type >::in_avail std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< char std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sputc(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

file()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
std::__c_file * __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::file ()  [inline]
```

Returns

The underlying FILE*.

This function can be used to access the underlying C file pointer. Note that there is no way for the library to track what you do with the file, so be careful.

gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_streambuf< char_type, traits_type >::uflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const  [inline], [inherited]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.
Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy
std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_stringbuf< _CharT,
std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< cha
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_streambuf< char_type, traits_type >::sgetc(),
std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< char >::showmanyc(), std::basic_streambuf< char_type, traits
std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_streambuf< char_type, traits_type >::uflow(),
std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(),
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and
xsgetn().

imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
            const locale & __loc)  [inline], [protected], [virtual], [inherited]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, and std::basic_filebuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

in_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ()  [inline], [inherited]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

overflow()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::overflow (
            int_type __c = traits_type::eof())  [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

_↩ _c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually __c, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if __c is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

pbackfail()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::pbackfail (
            int_type __c = traits_type::eof())  [inline], [protected], [virtual]
```

Tries to back up the input sequence.

Parameters

_↩ _c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

pbase()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, std::basic_stringbuf< char >::str(), std::basic_filebuf< _CharT, _Traits >::sync(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), and std::basic_streambuf< char_type, traits_type >::sputc().

pptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_streambuf< char_t std::basic_filebuf< _CharT, _Traits >::sync(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
            const locale & __loc)  [inline], [inherited]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
            off_type __off,
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ()  [inline], [inherited]
```

Calls virtual sync function.
Referenced by std::basic_istream< _CharT, _Traits >::sync().

sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_streambuf< char_type, traits_type >::snextc().

seekoff()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
            off_type ,
            ios_base::seekdir ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Ti std::basic_stringbuf< char >, and std::basic_stringbuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubseekoff().

seekpos()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (
            pos_type ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Ti std::basic_stringbuf< char >, and std::basic_stringbuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubseekpos().

setbuf()

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf< char_type, _Traits > * std::basic_streambuf< _CharT, _Traits >::setbuf
(
            char_type * ,
            streamsize )  [inline], [protected], [virtual], [inherited]
```

Manipulates the buffer.
Each derived class provides its own appropriate behavior. See the next-to-last paragraph of https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering for more on this function.

Note

Base class version does nothing, returns `this`.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, and std::basic_stringbuf< char >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubsetbuf().

setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
            char_type * __gbeg,
            char_type * __gnext,
            char_type * __gend)  [inline], [protected], [inherited]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), std::basic_filebuf< char_type, traits_type >::_M_create
std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_filebuf< char_type, traits_type >::_M_set_buffer(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
            char_type * __pbeg,
            char_type * __pend)  [inline], [protected], [inherited]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), and std::basic_filebuf< char_type, traits_type >::_M_s

sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::igr and std::basic_streambuf< char_type, traits_type >::snextc().

sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
             char_type * __s,
             streamsize __n)  [inline], [inherited]
```

Entry point for xsgetn.

Parameters

__s	A buffer area.
__n	A count.

Returns xsgetn(__s,__n). The effect is to fill __*s*[0] through __*s*[__n-1] with characters from the input sequence, if possible.

showmanyc()

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf< _CharT, _Traits >::showmanyc ()  [inline], [protected],
[virtual], [inherited]
```

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, and std::basic_stringbuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::in_avail().

snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, char_traits< _CharT > >::get(), and std::basic_istream< _CharT, char_traits< _CharT > >::readsome().

sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
            char_type __c)  [inline], [inherited]
```

Pushing characters back into the input stream.

Parameters

__c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but *__c* is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be *__c*.
Referenced by std::basic_istream< _CharT, _Traits >::putback().

sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
            char_type __c)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _c	A character to output.

Returns

__c, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __c in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.
Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow().

sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
            const char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _s	A buffer read area.
_↩ _n	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __s[0] through __s[__n-1] to the output sequence, if possible.

sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ()  [inline], [inherited]
```

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.
Referenced by std::basic_istream< _CharT, _Traits >::unget().

sync()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::sync ()  [inline], [protected],
[virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

uflow()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::uflow ()  [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.
Reimplemented from std::basic_streambuf< _CharT, _Traits >.

underflow()

```
template<typename _CharT, typename _Traits = std::char_traits<_CharT>>
virtual int_type __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >::underflow ()  [inline], [protected],
[virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

xsgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsgetn (
            char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character extraction.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __s[0] through __s[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __n characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, and std::basic_filebuf< char_type, traits_type >.
References egptr(), gptr(), std::min(), and uflow().
Referenced by std::basic_streambuf< char_type, traits_type >::sgetn().

xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character insertion.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __s[0] through __s[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, and std::basic_filebuf< char_type, traits_type >.
References epptr(), std::min(), overflow(), and pptr().
Referenced by std::basic_streambuf< char_type, traits_type >::sputn().

6.931.3 Member Data Documentation

_M_buf_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale  [protected], [inherited]
```

Current locale setting.
Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

_M_in_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg  [protected], [inherited]
```

Start of get area.

_M_in_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur  [protected], [inherited]
```

Current read area.

_M_in_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end  [protected], [inherited]
```

End of get area.

_M_out_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg  [protected], [inherited]
```

Start of put area.

_M_out_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur  [protected], [inherited]
```

Current put area.

_M_out_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end  [protected], [inherited]
```

End of put area.
The documentation for this class was generated from the following file:

- stdio_sync_filebuf.h

6.932 std::chrono::steady_clock Struct Reference

```
#include <chrono.h>
```

Public Types

- typedef chrono::nanoseconds **duration**
- typedef duration::period **period**

- typedef duration::rep **rep**
- typedef chrono::time_point< steady_clock, duration > **time_point**

Static Public Member Functions

- static time_point **now** () noexcept

Static Public Attributes

- static constexpr bool **is_steady**

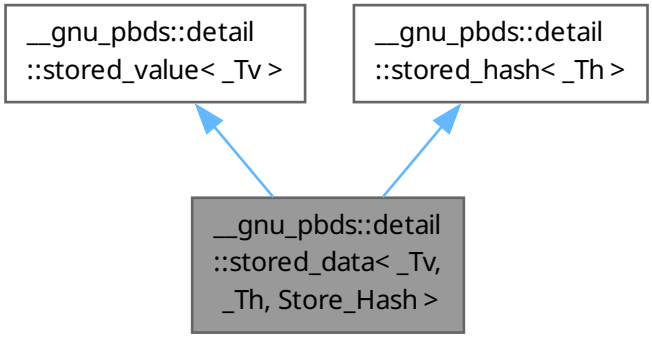
6.932.1 Detailed Description

Monotonic clock.
Time returned has the property of only increasing at a uniform rate.
The documentation for this struct was generated from the following file:

- chrono.h

6.933 __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash > Struct Template Reference

`#include <types_traits.hpp>`
Inheritance diagram for __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >:



Public Types

- typedef _Th **hash_type**
- typedef _Tv **value_type**

Public Attributes

- hash_type **m_hash**
- value_type **m_value**

6.933.1 Detailed Description

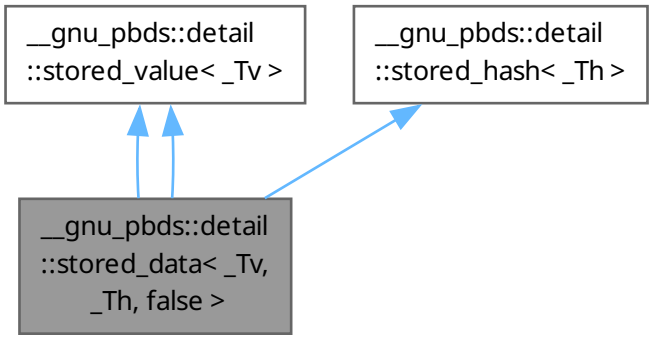
template<typename _Tv, typename _Th, bool Store_Hash>
struct __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >

Primary template for representation of stored data. Two types of data can be stored: value and hash.
The documentation for this struct was generated from the following file:

- types_traits.hpp

6.934 __gnu_pbds::detail::stored_data< _Tv, _Th, false > Struct Template Reference

`#include <types_traits.hpp>`
Inheritance diagram for __gnu_pbds::detail::stored_data< _Tv, _Th, false >:



Public Types

- typedef _Th **hash_type**
- typedef _Tv **value_type**

Public Attributes

- hash_type **m_hash**
- value_type **m_value**

6.934.1 Detailed Description

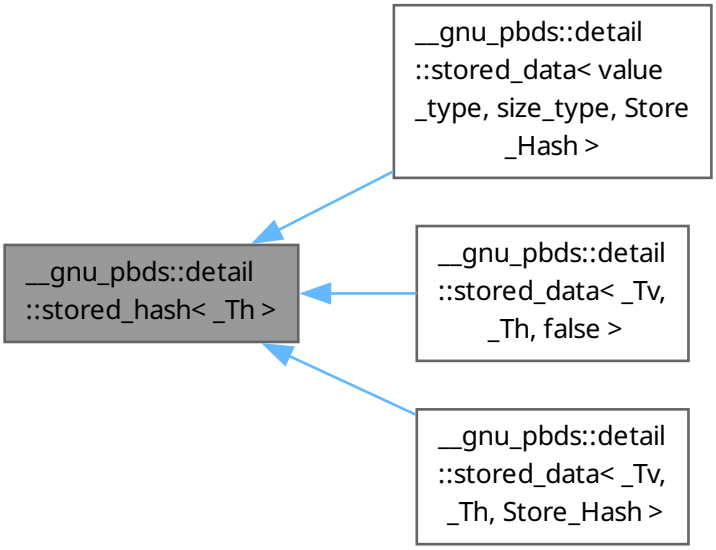
template<typename _Tv, typename _Th>
struct __gnu_pbds::detail::stored_data< _Tv, _Th, false >

Specialization for representation of stored data of just value type.
The documentation for this struct was generated from the following file:

- types_traits.hpp

6.935 __gnu_pbds::detail::stored_hash< _Th > Struct Template Reference

`#include <types_traits.hpp>`
Inheritance diagram for __gnu_pbds::detail::stored_hash< _Th >:



Public Types

- typedef _Th **hash_type**

Public Attributes

- hash_type **m_hash**

6.935.1 Detailed Description

template<**typename _Th**>
struct __gnu_pbds::detail::stored_hash< **_Th** >

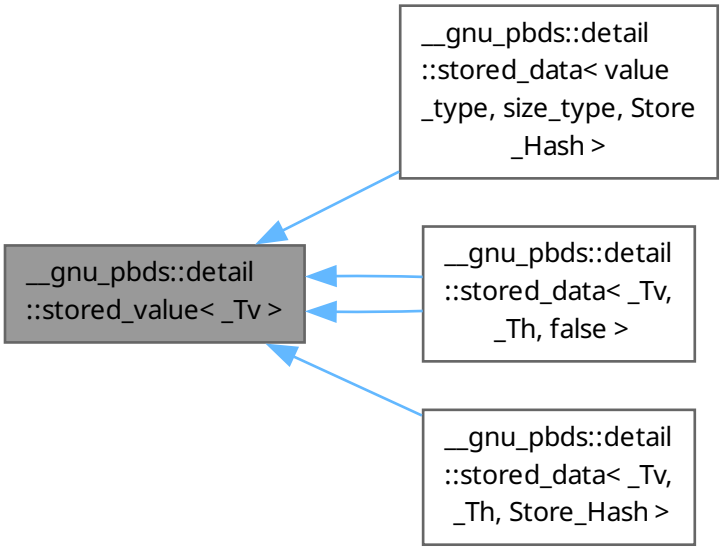
Stored hash.
The documentation for this struct was generated from the following file:

- types_traits.hpp

6.936 __gnu_pbds::detail::stored_value< _Tv > Struct Template Reference

`#include <types_traits.hpp>`

Inheritance diagram for __gnu_pbds::detail::stored_value< _Tv >:



Public Types

- typedef _Tv **value_type**

Public Attributes

- value_type **m_value**

6.936.1 Detailed Description

template<**typename _Tv**>
struct __gnu_pbds::detail::stored_value< **_Tv** >

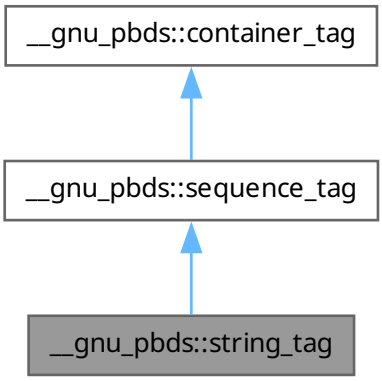
Stored value.
The documentation for this struct was generated from the following file:

- types_traits.hpp

6.937 __gnu_pbds::string_tag Struct Reference

`#include <tag_and_trait.hpp>`

Inheritance diagram for __gnu_pbds::string_tag:



6.937.1 Detailed Description

Basic string container, inclusive of strings, ropes, etc.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.938 std::student_t_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **student_t_distribution** (_RealType __n)
- **student_t_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)

- result_type max () const
- result_type min () const
- _RealType **n** () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::student_t_distribution< _RealType1 > &__x)
- bool operator== (const student_t_distribution &__d1, const student_t_distribution &__d2)
- template<typename _RealType1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::student_t_distribution< _RealType1 > &__x)

6.938.1 Detailed Description

template<typename _RealType = double>
class std::student_t_distribution< _RealType >

A student_t_distribution random number distribution.
The formula for the normal probability mass function is:

$$p(x|n) = \frac{1}{\sqrt{(n\pi)}} \frac{\Gamma((n+1)/2)}{\Gamma(n/2)} (1 + \frac{x^2}{n})^{-(n+1)/2}$$

Since

C++11

6.938.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::student_t_distribution< _RealType >::result_type
```

The type of the range of the distribution.

6.938.3 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::student_t_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::student_t_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::student_t_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.

param() [1/2]

```
template<typename _RealType = double>
param_type std::student_t_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.

param() [2/2]

```
template<typename _RealType = double>
void std::student_t_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::student_t_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

6.938.4 Friends And Related Symbol Documentation

operator<<

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::student_t_distribution< _RealType1 > & __x)  [friend]
```

Inserts a student_t_distribution random number distribution __x into the output stream __os.

Parameters

__os	An output stream.
__x	A student_t_distribution random number distribution.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _RealType = double>
bool operator== (
             const student_t_distribution< _RealType > & __d1,
             const student_t_distribution< _RealType > & __d2)  [friend]
```

Return true if two Student t distributions have the same parameters and the sequences that would be generated are equal.

operator>>

```
template<typename _RealType = double>
template<typename _RealType1, typename _CharT, typename _Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
             std::basic_istream< _CharT, _Traits > & __is,
             std::student_t_distribution< _RealType1 > & __x)  [friend]
```

Extracts a student_t_distribution random number distribution __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A student_t_distribution random number generator engine.

Returns

The input stream with __x extracted or in an error state.

The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.939 std::sub_match< _BiIter > Class Template Reference

```
#include <regex>
```

Public Types

- typedef __iter_traits::difference_type **difference_type**
- typedef _BiIter **iterator**
- typedef basic_string< value_type > **string_type**
- typedef __iter_traits::value_type **value_type**

Public Member Functions

- int compare (const sub_match &__s) const
- difference_type length () const noexcept
- operator string_type () const
- string_type str () const
- void swap (sub_match &__s) noexcept(__is_nothrow_swappable< _BiIter >::value)

- int compare (const string_type &__s) const
- int compare (const value_type *__s) const

Public Attributes

- iterator **first**
- bool **matched**
- iterator **second**

Related Symbols

(Note that these are not member symbols.)

- template<typename _BiIter>
  bool operator== (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs)
- template<typename _BiIter>
  auto operator<=> (const sub_match< _BiIter > &__lhs, const sub_match< _BiIter > &__rhs) noexcept(__↩
  detail::__is_contiguous_iter< _BiIter >::value)
- template<typename _Bi_iter, typename _Ch_traits, typename _Ch_alloc>
  bool operator== (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _Ch↩
  _alloc > &__rhs)
- template<typename _Bi_iter, typename _Ch_traits, typename _Alloc>
  auto operator<=> (const sub_match< _Bi_iter > &__lhs, const __sub_match_string< _Bi_iter, _Ch_traits, _↩
  Alloc > &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)
- template<typename _Bi_iter>
  bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const
  *__rhs)
- template<typename _Bi_iter>
  auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const
  *__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)
- template<typename _Bi_iter>
  bool operator== (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const
  &__rhs)
- template<typename _Bi_iter>
  auto operator<=> (const sub_match< _Bi_iter > &__lhs, typename iterator_traits< _Bi_iter >::value_type const
  &__rhs) noexcept(__detail::__is_contiguous_iter< _Bi_iter >::value)
- template<typename _Ch_type, typename _Ch_traits, typename _Bi_iter>
  basic_ostream< _Ch_type, _Ch_traits > & operator<< (basic_ostream< _Ch_type, _Ch_traits > &__os, const
  sub_match< _Bi_iter > &__m)

6.939.1 Detailed Description

template<typename _BiIter>
class std::sub_match< _BiIter >

A sequence of characters matched by a particular marked sub-expression.
An object of this class is essentially a pair of iterators marking a matched subexpression within a regular expression pattern match. Such objects can be converted to and compared with std::basic_string objects of the same character type as the pattern matched by the regular expression.
A `sub_match<Iter>` has a public base class of type `pair<Iter, Iter>`, so inherits pair's data members named `first` and `second`. The iterators that make up the pair are the usual half-open interval referencing the actual original pattern matched.

Since

C++11

6.939.2 Member Function Documentation

compare() [1/3]

```
template<typename _BiIter>
int std::sub_match< _BiIter >::compare (
            const string_type & __s) const  [inline]
```

Compares this sub_match to a string.

Parameters

_↩ _s	A string to compare to this sub_match.

Return values

negative	This matched sequence will collate before __s.
zero	This matched sequence is equivalent to __s.
positive	This matched sequence will collate after __s.

compare() [2/3]

```
template<typename _BiIter>
int std::sub_match< _BiIter >::compare (
            const sub_match< _BiIter > & __s) const  [inline]
```

Compares this and another matched sequence.

Parameters

_↩ _s	Another matched sequence to compare to this one.

Return values

negative	This matched sequence will collate before __s.
zero	This matched sequence is equivalent to __s.
positive	This matched sequence will collate after __s.

Referenced by std::sub_match< const char * >::operator==(), and std::sub_match< const char * >::operator==().

compare() [3/3]

```
template<typename _BiIter>
int std::sub_match< _BiIter >::compare (
            const value_type * __s) const  [inline]
```

Compares this sub_match to a string.

Parameters

_↩ _s	A string to compare to this sub_match.

Return values

negative	This matched sequence will collate before __s.
zero	This matched sequence is equivalent to __s.
positive	This matched sequence will collate after __s.

length()

```
template<typename _BiIter>
difference_type std::sub_match< _BiIter >::length () const  [inline], [noexcept]
```

Gets the length of the matching sequence.

operator string_type()

```
template<typename _BiIter>
std::sub_match< _BiIter >::operator string_type () const  [inline]
```

Gets the matching sequence as a string.

Returns

the matching sequence as a string.

This is the implicit conversion operator. It is identical to the str() member function except that it will want to pop up in unexpected places and cause a great deal of confusion and cursing from the unwary.

str()

```
template<typename _BiIter>
string_type std::sub_match< _BiIter >::str () const  [inline]
```

Gets the matching sequence as a string.

Returns

the matching sequence as a string.

Referenced by std::sub_match< const char * >::operator string_type().

swap()

```
template<typename _BiIter>
void std::sub_match< _BiIter >::swap (
            sub_match< _BiIter > & __s)  [inline], [noexcept]
```

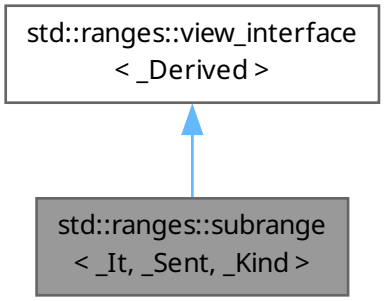
Swap the values of two sub_match objects.

The documentation for this class was generated from the following file:

- regex.h

6.940 std::ranges::subrange< _It, _Sent, _Kind > Class Template Reference

`#include <ranges_util.h>`
Inheritance diagram for std::ranges::subrange< _It, _Sent, _Kind >:



Public Member Functions

- constexpr **subrange** (__detail::__convertible_to_non_slicing< _It > auto __i, _Sent __s) noexcept(is_nothrow_constructible_v< _It, decltype(__i)> &&is_nothrow_constructible_v< _Sent, _Sent & >)
- constexpr **subrange** (__detail::__convertible_to_non_slicing< _It > auto __i, _Sent __s, __size_type __n) noexcept(is_nothrow_constructible_v< _It, decltype(__i)> &&is_nothrow_constructible_v< _Sent, _Sent & >)
- template<__detail::__different_from< subrange > _Rng>
  requires _S_store_size
  constexpr **subrange** (_Rng &&__r) noexcept(noexcept(subrange(__r, ranges::size(__r)))) &&sized_range< _Rng >
- template<__detail::__different_from< subrange > _Rng>
  requires (!_S_store_size)
  constexpr **subrange** (_Rng &&__r) noexcept(noexcept(subrange(ranges::begin(__r), ranges::end(__r))))
- template<borrowed_range _Rng>
  requires (_Kind == subrange_kind::sized)
  constexpr **subrange** (_Rng &&__r, __size_type __n) noexcept(noexcept(subrange(ranges::begin(__r), ranges::end(__r), __n)))
- constexpr subrange & **advance** (iter_difference_t< _It > __n)
- constexpr decltype(auto) **back** ()
- constexpr decltype(auto) **back** () const
- constexpr _It **begin** ()
- constexpr _It **begin** () const
- constexpr auto **cbegin** ()
- constexpr auto **cbegin** () const
- constexpr auto **cend** ()
- constexpr auto **cend** () const
- constexpr auto **data** () const noexcept(noexcept(ranges::begin(_M_derived())))
- constexpr auto **data** () noexcept(noexcept(ranges::begin(_M_derived())))
- constexpr bool **empty** () const
- constexpr bool **empty** () noexcept(noexcept(_S_empty(_M_derived())))

- constexpr _Sent **end** () const
- constexpr decltype(auto) **front** ()
- constexpr decltype(auto) **front** () const
- constexpr subrange **next** (iter_difference_t< _It > __n=1) &&
- constexpr subrange **next** (iter_difference_t< _It > __n=1) const &
- template<__detail::__different_from< subrange > _PairLike>
  constexpr **operator _PairLike** () const
- constexpr **operator bool** () const noexcept(noexcept(ranges::empty(_M_derived())))
- constexpr **operator bool** () noexcept(noexcept(ranges::empty(_M_derived())))
- template<random_access_range _Range = _Derived>
  constexpr decltype(auto) **operator[ ]** (range_difference_t< _Range > __n)
- template<random_access_range _Range = const _Derived>
  constexpr decltype(auto) **operator[ ]** (range_difference_t< _Range > __n) const
- constexpr subrange **prev** (iter_difference_t< _It > __n=1) const
- constexpr __size_type **size** () const
- constexpr auto **size** () noexcept(noexcept(_S_size(_M_derived())))

Friends

- struct **views::_Drop**

6.940.1 Detailed Description

template<input_or_output_iterator _It, sentinel_for< _It > _Sent = _It, subrange_kind _Kind = sized_sentinel↩ _for<_Sent, _It> ? subrange_kind::sized : subrange_kind::unsized>
requires (_Kind == subrange_kind::sized || !sized_sentinel_for<_Sent, _It>)
class std::ranges::subrange< _It, _Sent, _Kind >

The ranges::subrange class template.
The documentation for this class was generated from the following file:

- ranges_util.h

6.941 std::subtract_with_carry_engine< _UIntType, __w, __s, __r > Class Template Reference

`#include <random>`

Public Types

- typedef _UIntType result_type

Public Member Functions

- template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
  subtract_with_carry_engine (_Sseq &__q)
- subtract_with_carry_engine (result_type __sd)
- void discard (unsigned long long __z)
- result_type operator() ()
- template<typename _Sseq>
  _If_seed_seq< _Sseq > seed (_Sseq &__q)
- void seed (result_type __sd=0u)

Static Public Member Functions

- static constexpr result_type max ()
- static constexpr result_type min ()

Static Public Attributes

- static constexpr uint_least32_t **default_seed**
- static constexpr size_t **long_lag**
- static constexpr size_t **short_lag**
- static constexpr size_t **word_size**

Friends

- template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > &__x)
- bool operator== (const subtract_with_carry_engine &__lhs, const subtract_with_carry_engine &__rhs)
- template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & operator>> (std::basic_istream< _CharT, _Traits > &__is, std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > &__x)

6.941.1 Detailed Description

template<typename _UIntType, size_t __w, size_t __s, size_t __r>
class std::subtract_with_carry_engine< _UIntType, __w, __s, __r >

The Marsaglia-Zaman generator.
This is a model of a Generalized Fibonacci discrete random number generator, sometimes referred to as the SWC generator.
A discrete random number generator that produces pseudorandom numbers using:

$$x_i \leftarrow (x_{i-s} - x_{i-r} - carry_{i-1}) \bmod m$$

The size of the state is r and the maximum period of the generator is $(m^r - m^s - 1)$.

Since

C++11

6.941.2 Member Typedef Documentation

result_type

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
typedef _UIntType std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::result_type
```

The type of the generated random value.

6.941.3 Constructor & Destructor Documentation

subtract_with_carry_engine() [1/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine (
            result_type __sd)  [inline], [explicit]
```

Constructs an explicitly seeded subtract_with_carry_engine random number generator.

subtract_with_carry_engine() [2/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _Sseq, typename = _If_seed_seq<_Sseq>>
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::subtract_with_carry_engine (
            _Sseq & __q)  [inline], [explicit]
```

Constructs a subtract_with_carry_engine random number engine seeded from the seed sequence __q.

Parameters

_↩ _q	the seed sequence.

6.941.4 Member Function Documentation

discard()

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
void std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::discard (
            unsigned long long __z)  [inline]
```

Discard a sequence of random numbers.

max()

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
constexpr result_type std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::max ()  [inline],
[static], [constexpr]
```

Gets the inclusive maximum value of the range of random integers returned by this generator.

min()

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
constexpr result_type std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::min ()  [inline],
[static], [constexpr]
```

Gets the inclusive minimum value of the range of random integers returned by this generator.

operator()()

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
subtract_with_carry_engine< _UIntType, __w, __s, __r >::result_type std::subtract_with_carry_engine<
_UIntType, __w, __s, __r >::operator() ()
```

Gets the next random number in the sequence.

seed() [1/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _Sseq>
auto std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::seed (
            _Sseq & __q) -> _If_seed_seq< _Sseq >
```

Seeds the initial state x_0 of the % subtract_with_carry_engine random number generator.

seed() [2/2]

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
```

```
void std::subtract_with_carry_engine< _UIntType, __w, __s, __r >::seed (
            result_type __sd = 0u)
```

Seeds the initial state x_0 of the random number generator.
N1688[4.19] modifies this as follows. If __value == 0, sets value to 19780503. In any case, with a linear congruential generator lcg(i) having parameters $m_{lcg} = 2147483563, a_{lcg} = 40014, c_{lcg} = 0, and lcg(0) = value$, sets $x_{-r} \dots x_{-1}$ to $lcg(1) \bmod m \dots lcg(r) \bmod m$ respectively. If $x_{-1} = 0$ set carry to 1, otherwise sets carry to 0.
Referenced by std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::subtract_with_carry_engine(), and std::subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 >::subtract_with_carry_engine().

6.941.5 Friends And Related Symbol Documentation

operator<<

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _↩
Traits>
std::basic_ostream< _CharT, _Traits > & operator<< (
            std::basic_ostream< _CharT, _Traits > & __os,
            const std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > & __x)  [friend]
```

Inserts the current state of a % subtract_with_carry_engine random number generator engine __x into the output stream __os.

Parameters

__os	An output stream.
__x	A % subtract_with_carry_engine random number generator engine.

Returns

The output stream with the state of __x inserted or in an error state.

operator==

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
bool operator== (
            const subtract_with_carry_engine< _UIntType, __w, __s, __r > & __lhs,
            const subtract_with_carry_engine< _UIntType, __w, __s, __r > & __rhs)  [friend]
```

Compares two % subtract_with_carry_engine random number generator objects of the same type for equality.

Parameters

__lhs	A % subtract_with_carry_engine random number generator object.
__rhs	Another % subtract_with_carry_engine random number generator object.

Returns

true if the infinite sequences of generated values would be equal, false otherwise.

operator>>

```
template<typename _UIntType, size_t __w, size_t __s, size_t __r>
template<typename _UIntType1, size_t __w1, size_t __s1, size_t __r1, typename _CharT, typename _↩
Traits>
std::basic_istream< _CharT, _Traits > & operator>> (
            std::basic_istream< _CharT, _Traits > & __is,
            std::subtract_with_carry_engine< _UIntType1, __w1, __s1, __r1 > & __x)  [friend]
```

Extracts the current state of a % subtract_with_carry_engine random number generator engine __x from the input stream __is.

Parameters

_↩ _is	An input stream.
_↩ _x	A % subtract_with_carry_engine random number generator engine.

Returns

The input stream with the state of __x extracted or in an error state.

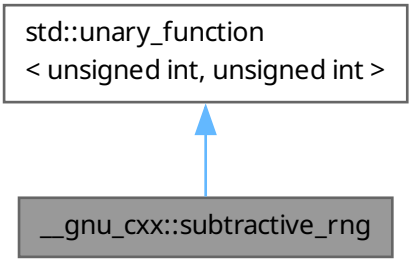
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.942 __gnu_cxx::subtractive_rng Class Reference

`#include <functional>`

Inheritance diagram for __gnu_cxx::subtractive_rng:



Public Types

- typedef unsigned int argument_type
- typedef unsigned int result_type

Public Member Functions

- subtractive_rng ()
- subtractive_rng (unsigned int __seed)
- void **_M_initialize** (unsigned int __seed)
- unsigned int operator() (unsigned int __limit)

6.942.1 Detailed Description

The `subtractive_rng` class is documented on `SGI's site`. Note that this code assumes that `int` is 32 bits.

6.942.2 Member Typedef Documentation

argument_type

```
typedef unsigned int std::unary_function< unsigned int, unsigned int >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef unsigned int std::unary_function< unsigned int, unsigned int >::result_type  [inherited]
```

`result_type` is the return type

6.942.3 Constructor & Destructor Documentation

subtractive_rng() [1/2]

```
__gnu_cxx::subtractive_rng::subtractive_rng (
            unsigned int __seed)  [inline]
```

Ctor allowing you to initialize the seed.

subtractive_rng() [2/2]

```
__gnu_cxx::subtractive_rng::subtractive_rng ()  [inline]
```

Default ctor; initializes its state with some number you don't see.

6.942.4 Member Function Documentation

operator()()

```
unsigned int __gnu_cxx::subtractive_rng::operator() (
            unsigned int __limit)  [inline]
```

Returns a number less than the argument.
The documentation for this class was generated from the following file:

- ext/functional

6.943 __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits > Struct Template Reference

`#include <synth_access_traits.hpp>`

Public Types

- typedef _ATraits **base_type**
- typedef base_type::const_iterator **const_iterator**
- typedef type_traits::const_reference **const_reference**

- typedef type_traits::key_const_reference **key_const_reference**
- typedef Type_Traits **type_traits**

Public Member Functions

- **synth_access_traits** (const base_type &)
- bool **cmp_keys** (key_const_reference, key_const_reference) const
- bool **cmp_prefixes** (const_iterator, const_iterator, const_iterator, const_iterator, bool compare_after=false) const
- bool **equal_keys** (key_const_reference, key_const_reference) const
- bool **equal_prefixes** (const_iterator, const_iterator, const_iterator, const_iterator, bool compare_after=true) const

Static Public Member Functions

- static key_const_reference **extract_key** (const_reference)

6.943.1 Detailed Description

template<typename Type_Traits, bool Set, typename _ATraits>
struct __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >

Synthetic element access traits.
The documentation for this struct was generated from the following file:

- synth_access_traits.hpp

6.944 std::chrono::system_clock Struct Reference

`#include <chrono.h>`

Public Types

- typedef chrono::nanoseconds **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**
- typedef chrono::time_point< system_clock, duration > **time_point**

Static Public Member Functions

- static time_point **from_time_t** (std::time_t __t) noexcept
- static time_point **now** () noexcept
- static std::time_t **to_time_t** (const time_point &__t) noexcept

Static Public Attributes

- static constexpr bool **is_steady**

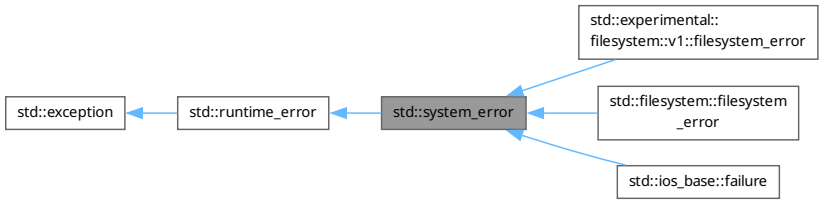
6.944.1 Detailed Description

System clock.
Time returned represents wall time from the system-wide clock.
The documentation for this struct was generated from the following file:

- chrono.h

6.945 std::system_error Class Reference

`#include <system_error>`
Inheritance diagram for std::system_error:



Public Member Functions

- **system_error** (const system_error &)=default
- **system_error** (error_code __ec, const char *__what)
- **system_error** (error_code __ec, const string &__what)
- **system_error** (error_code __ec=error_code())
- **system_error** (int __v, const error_category &__ecat)
- **system_error** (int __v, const error_category &__ecat, const char *__what)
- **system_error** (int __v, const error_category &__ecat, const string &__what)
- const error_code & **code** () const noexcept
- system_error & **operator=** (const system_error &)=default
- virtual const char * what () const noexcept

6.945.1 Detailed Description

An exception type that includes an `error_code` value.
Typically used to report errors from the operating system and other low-level APIs.

Since

C++11

6.945.2 Member Function Documentation

what()

`virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]`
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::experimental::filesystem::v1::filesystem_error, std::filesystem::filesystem_error, and std::ios_base::failure.
References what().
Referenced by what().
The documentation for this class was generated from the following file:

- system_error

6.946 std::chrono::tai_clock Class Reference

`#include <chrono>`

Public Types

- using **duration**
- using **period**
- using **rep**
- using **time_point**

Static Public Member Functions

- template<typename _Duration>
  static tai_time< common_type_t< _Duration, seconds > > **from_utc** (const utc_time< _Duration > &__t)
- static time_point **now** ()
- template<typename _Duration>
  static utc_time< common_type_t< _Duration, seconds > > **to_utc** (const tai_time< _Duration > &__t)

Static Public Attributes

- static constexpr bool **is_steady**

6.946.1 Detailed Description

A clock that measures International Atomic Time.
The epoch is 1958-01-01 00:00:00.

Since

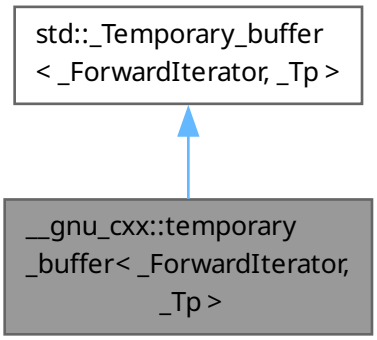
C++20

The documentation for this class was generated from the following file:

- chrono

6.947 __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp > Struct Template Reference

`#include <memory>`
Inheritance diagram for __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >:



Public Types

- typedef pointer **iterator**
- typedef value_type * **pointer**
- typedef ptrdiff_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- temporary_buffer (_ForwardIterator __first, _ForwardIterator __last)
- ~temporary_buffer ()
- size_type _M_requested_size () const
- iterator begin ()
- iterator end ()
- size_type size () const

Protected Attributes

- struct std::_Temporary_buffer::_Impl **_M_impl**
- size_type **_M_original_len**

6.947.1 Detailed Description

template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::value_type>
struct __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >

This class provides similar behavior and semantics of the standard functions get_temporary_buffer() and return_↩ temporary_buffer(), but encapsulated in a type vaguely resembling a standard container.
By default, a temporary_buffer<Iter> stores space for objects of whatever type the Iter iterator points to. It is constructed from a typical [first,last) range, and provides the begin(), end(), size() functions, as well as requested_size(). For non-trivial types, copies of *first will be used to initialize the storage.
`malloc` is used to obtain underlying storage.
Like get_temporary_buffer(), not all the requested memory may be available. Ideally, the created buffer will be large enough to hold a copy of [first,last), but if size() is less than requested_size(), then this didn't happen.

6.947.2 Constructor & Destructor Documentation

temporary_buffer()

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::↩
value_type>
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::temporary_buffer (
            _ForwardIterator __first,
            _ForwardIterator __last)  [inline]
```

Requests storage large enough to hold a copy of [first,last).
References std::_Temporary_buffer< _ForwardIterator, _Tp >::_Temporary_buffer(), and __gnu_cxx::distance().

~temporary_buffer()

```
template<class _ForwardIterator, class _Tp = typename std::iterator_traits<_ForwardIterator>::↩
value_type>
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >::~temporary_buffer ()  [inline]
```

Destroys objects and frees storage.

6.947.3 Member Function Documentation

_M_requested_size()

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::_M_requested_size () const  [inline],
[inherited]
```

Returns the size requested by the constructor; may be >size().

begin()

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::begin ()  [inline], [inherited]
```

As per Table mumble.
Referenced by _Temporary_buffer().

end()

```
template<typename _ForwardIterator, typename _Tp>
iterator std::_Temporary_buffer< _ForwardIterator, _Tp >::end ()  [inline], [inherited]
```

As per Table mumble.
Referenced by _Temporary_buffer().

size()

```
template<typename _ForwardIterator, typename _Tp>
size_type std::_Temporary_buffer< _ForwardIterator, _Tp >::size () const  [inline], [inherited]
```

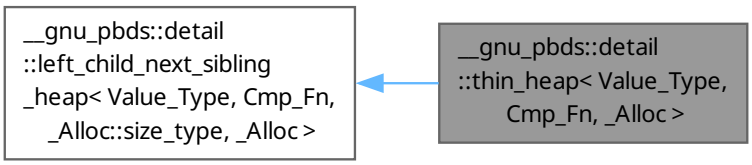
As per Table mumble.
The documentation for this struct was generated from the following file:

- ext/memory

6.948 __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc > Class Template Reference

```
#include <thin_heap_.hpp>
```

Inheritance diagram for __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >:


__gnu_pbds::detail
::left_child_next_sibling
_heap< Value_Type, Cmp_Fn,
_Alloc::size_type, _Alloc >
__gnu_pbds::detail
::thin_heap< Value_Type,
Cmp_Fn

- typedef __rebind_a::const_pointer **const_pointer**
- typedef __rebind_a::const_reference **const_reference**
- typedef _Alloc::difference_type **difference_type**
- typedef base_type::iterator **iterator**
- typedef base_type::point_const_iterator **point_const_iterator**
- typedef base_type::point_iterator **point_iterator**
- typedef __rebind_a::pointer **pointer**
- typedef __rebind_a::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef Value_Type **value_type**

Public Member Functions

- iterator **begin** ()
- void **clear** ()
- bool **empty** () const
- iterator **end** ()
- void **erase** (point_iterator)
- template<typename Pred>
  size_type **erase_if** (Pred)
- Cmp_Fn & **get_cmp_fn** ()
- void **join** (thin_heap< Value_Type, Cmp_Fn, _Alloc > &)
- size_type **max_size** () const
- void **modify** (point_iterator, const_reference)
- void **pop** ()
- point_iterator **push** (const_reference)
- size_type **size** () const
- template<typename Pred>
  void **split** (Pred, thin_heap< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap** (left_child_next_sibling_heap< Value_Type, Cmp_Fn, _Alloc::size_type, _Alloc > &)
- const_reference **top** () const

Protected Types

- typedef base_type::node **node**
- typedef alloc_traits::allocator_type **node_allocator**
- typedef base_type::node_const_pointer **node_const_pointer**
- typedef _Alloc::size_type **node_metadata**
- typedef base_type::node_pointer **node_pointer**
- typedef std::pair< node_pointer, node_pointer > **node_pointer_pair**

Protected Member Functions

- **thin_heap** (const Cmp_Fn &)
- **thin_heap** (const thin_heap< Value_Type, Cmp_Fn, _Alloc > &)
- void **actual_erase_node** (node_pointer)
- void **bubble_to_top** (node_pointer)
- void **clear_imp** (node_pointer)
- template<typename It>
  void **copy_from_range** (It, It)
- node_pointer **get_new_node_for_insert** (const_reference)
- node_pointer **prune** (Pred)

- void **swap** (thin_heap< Value_Type, Cmp_Fn, _Alloc > &)
- void **swap_with_parent** (node_pointer, node_pointer)
- void **to_linked_list** ()
- void **value_swap** (left_child_next_sibling_heap &)

Static Protected Member Functions

- static node_pointer **parent** (node_pointer)

Protected Attributes

- node_pointer **m_p_root**
- size_type **m_size**

6.948.1 Detailed Description

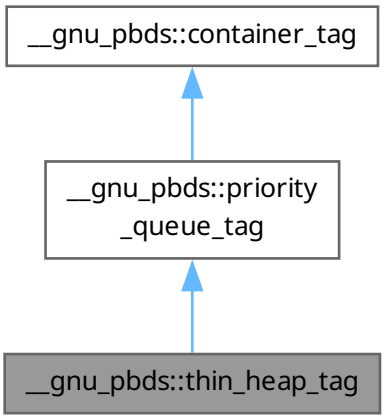
template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >

Thin heap.
See Tarjan and Kaplan.
The documentation for this class was generated from the following file:

- thin_heap_.hpp

6.949 __gnu_pbds::thin_heap_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::thin_heap_tag:



6.949.1 Detailed Description

Thin heap.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.950 std::thread Class Reference

`#include <thread>`

Classes

- class id

Public Types

- using **native_handle_type**

Public Member Functions

- template<typename _Callable, typename... _Args, typename = _Require<__not_same<_Callable>>>
  thread (_Callable &&__f, _Args &&... __args)
- **thread** (const thread &)=delete
- **thread** (thread &&__t) noexcept
- void **detach** ()
- id **get_id** () const noexcept
- void **join** ()
- bool **joinable** () const noexcept
- native_handle_type native_handle ()
- thread & **operator=** (const thread &)=delete
- thread & **operator=** (thread &&__t) noexcept
- void **swap** (thread &__t) noexcept

Static Public Member Functions

- static unsigned int **hardware_concurrency** () noexcept

Related Symbols

(Note that these are not member symbols.)

- bool **operator==** (thread::id __x, thread::id __y) noexcept
- void **swap** (thread &__x, thread &__y) noexcept

6.950.1 Detailed Description

A std::thread represents a new thread of execution.
The default constructor creates an object that does not own a thread. The `thread(F&&, Args&&...)` constructor invokes a callable in a new thread, and owns that new thread. A `std::thread` that owns a thread is *joinable*. Joining a thread waits for it to finish executing, which happens when the callable running in that thread returns.
A `std::thread` cannot be copied, but can be moved. Moving a joinable object transfers ownership of its thread to another object.
A joinable `std::thread` must be explicitly joined (or detached) before it is destroyed or assigned to. Attempting to destroy a joinable thread will terminate the whole process.

Since

C++11

6.950.2 Member Function Documentation

native_handle()

```
native_handle_type std::thread::native_handle ()  [inline]
```

Precondition

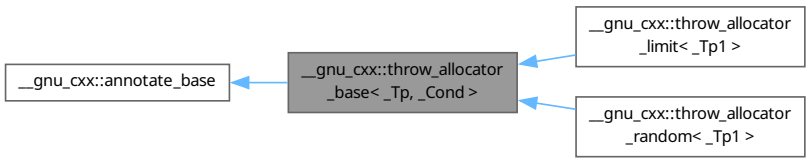
thread is joinable

The documentation for this class was generated from the following file:

- std_thread.h

6.951 __gnu_cxx::throw_allocator_base< _Tp, _Cond > Class Template Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::throw_allocator_base< _Tp, _Cond >:



Public Types

- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef value_type * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- const_pointer **address** (const_reference __x) const noexcept
- pointer **address** (reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *__hint=0)
- void **check** (size_type __n)
- map_alloc_type::iterator **check_allocated** (void *p, size_t size)
- void **check_allocated** (pointer __p, size_type __n)
- void **check_constructed** (size_t label)
- map_construct_type::iterator **check_constructed** (void *p)
- template<typename _Up, typename... _Args>
  void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- template<typename _Up>
  void **destroy** (_Up *__p)

- void **erase** (void *p, size_t size)
- void **erase_construct** (void *p)
- void **insert** (void *p, size_t size)
- void **insert_construct** (void *p)
- size_type **max_size** () const noexcept

Static Public Member Functions

- static void **check** ()
- static size_t **get_label** ()
- static void **set_label** (size_t l)

6.951.1 Detailed Description

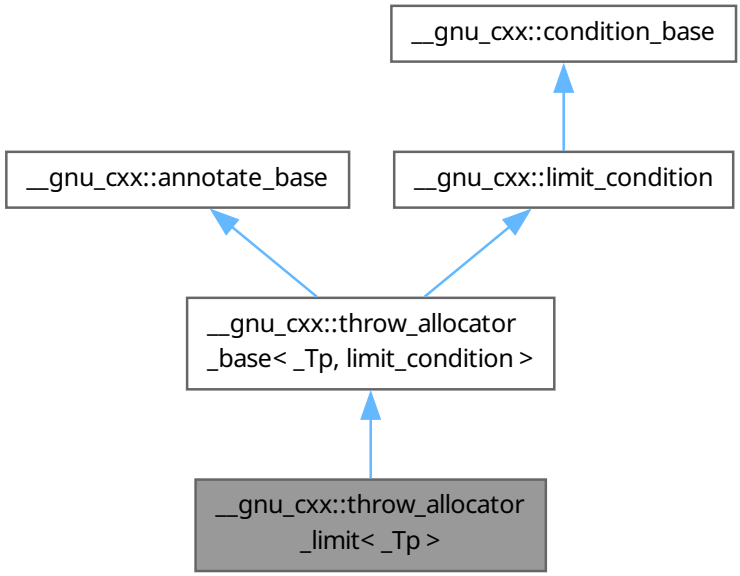
template<typename _Tp, typename _Cond>
class __gnu_cxx::throw_allocator_base< _Tp, _Cond >

Allocator class with logging and exception generation control. Intended to be used as an allocator_type in templatized code.
Note: Deallocate not allowed to throw.
The documentation for this class was generated from the following file:

- throw_allocator.h

6.952 __gnu_cxx::throw_allocator_limit< _Tp > Struct Template Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::throw_allocator_limit< _Tp >:



Public Types

- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef value_type * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **throw_allocator_limit** (const throw_allocator_limit &) noexcept
- template<typename _Tp1>
  throw_allocator_limit (const throw_allocator_limit< _Tp1 > &) noexcept
- pointer **address** (reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *__hint=0)
- void **check** (size_type __n)
- map_alloc_type::iterator **check_allocated** (void *p, size_t size)
- void **check_allocated** (pointer __p, size_type __n)
- void **check_constructed** (size_t label)
- map_construct_type::iterator **check_constructed** (void *p)
- void **construct** (_Up *__p, _Args &&... __args)
- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (_Up *__p)
- void **erase** (void *p, size_t size)
- void **erase_construct** (void *p)
- void **insert** (void *p, size_t size)
- void **insert_construct** (void *p)
- size_type **max_size** () const noexcept
- throw_allocator_limit & **operator=** (const throw_allocator_limit &)=default

Static Public Member Functions

- static void **check** ()
- static size_t & **count** ()
- static size_t **get_label** ()
- static size_t & **limit** ()
- static void **set_label** (size_t l)
- static void **set_limit** (const size_t __l)
- static void **throw_conditionally** ()

6.952.1 Detailed Description

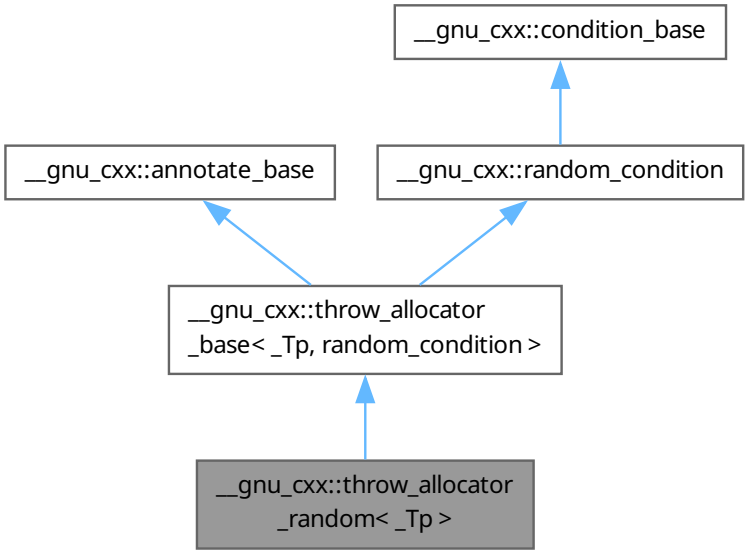
template<typename _Tp>
struct __gnu_cxx::throw_allocator_limit< _Tp >

Allocator throwing via limit condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.953 __gnu_cxx::throw_allocator_random< _Tp > Struct Template Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::throw_allocator_random< _Tp >:



Public Types

- typedef const value_type * **const_pointer**
- typedef const value_type & **const_reference**
- typedef std::ptrdiff_t **difference_type**
- typedef value_type * **pointer**
- typedef std::true_type **propagate_on_container_move_assignment**
- typedef value_type & **reference**
- typedef std::size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- **throw_allocator_random** (const throw_allocator_random &) noexcept
- template<typename _Tp1>
  throw_allocator_random (const throw_allocator_random< _Tp1 > &) noexcept
- pointer **address** (reference __x) const noexcept
- pointer **allocate** (size_type __n, const void *__hint=0)
- void **check** (size_type __n)
- void **check_allocated** (pointer __p, size_type __n)
- map_alloc_type::iterator **check_allocated** (void *p, size_t size)
- void **check_constructed** (size_t label)
- map_construct_type::iterator **check_constructed** (void *p)
- void **construct** (_Up *__p, _Args &&... __args)

- void **deallocate** (pointer __p, size_type __n)
- void **destroy** (_Up *__p)
- void **erase** (void *p, size_t size)
- void **erase_construct** (void *p)
- void **insert** (void *p, size_t size)
- void **insert_construct** (void *p)
- size_type **max_size** () const noexcept
- throw_allocator_random & **operator=** (const throw_allocator_random &)=default
- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **check** ()
- static size_t **get_label** ()
- static void **set_label** (size_t l)
- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

6.953.1 Detailed Description

template<typename _Tp>
struct __gnu_cxx::throw_allocator_random< _Tp >

Allocator throwing via random condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.954 __gnu_cxx::throw_value_base< _Cond > Struct Template Reference

`#include <throw_allocator.h>`

Public Types

- typedef _Cond **condition_type**

Public Member Functions

- **throw_value_base** (const std::size_t __i)
- **throw_value_base** (const throw_value_base &__v)
- **throw_value_base** (throw_value_base &&)=default
- throw_value_base & **operator++** ()
- throw_value_base & **operator=** (const throw_value_base &__v)
- throw_value_base & **operator=** (throw_value_base &&)=default

Public Attributes

- std::size_t **_M_i**

6.954.1 Detailed Description

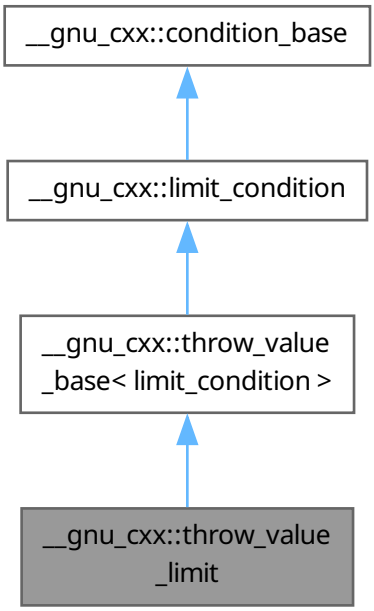
template<**typename _Cond**>
struct __gnu_cxx::throw_value_base< **_Cond** >

Class with exception generation control. Intended to be used as a value_type in templatized code.
Note: Destructor not allowed to throw.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.955 __gnu_cxx::throw_value_limit Struct Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::throw_value_limit:



Public Types

- typedef throw_value_base< limit_condition > **base_type**
- typedef limit_condition **condition_type**

Public Member Functions

- **throw_value_limit** (const std::size_t __i)
- **throw_value_limit** (const throw_value_limit &__other)
- **throw_value_limit** (throw_value_limit &&)=default
- throw_value_base & **operator++** ()
- throw_value_limit & **operator=** (const throw_value_limit &__other)
- throw_value_limit & **operator=** (throw_value_limit &&)=default

Static Public Member Functions

- static size_t & **count** ()
- static size_t & **limit** ()
- static void **set_limit** (const size_t __l)
- static void **throw_conditionally** ()

Public Attributes

- std::size_t **_M_i**

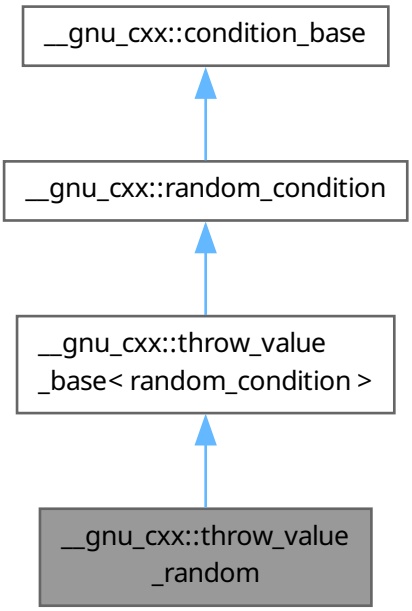
6.955.1 Detailed Description

Type throwing via limit condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.956 __gnu_cxx::throw_value_random Struct Reference

`#include <throw_allocator.h>`
Inheritance diagram for __gnu_cxx::throw_value_random:



Public Types

- typedef throw_value_base< random_condition > **base_type**
- typedef random_condition **condition_type**

Public Member Functions

- **throw_value_random** (const std::size_t __i)
- **throw_value_random** (const throw_value_random &__other)
- **throw_value_random** (throw_value_random &&)=default
- throw_value_base & **operator++** ()
- throw_value_random & **operator=** (const throw_value_random &__other)
- throw_value_random & **operator=** (throw_value_random &&)=default
- void **seed** (unsigned long __s)

Static Public Member Functions

- static void **set_probability** (double __p)
- static void **throw_conditionally** ()

Public Attributes

- std::size_t **_M_i**

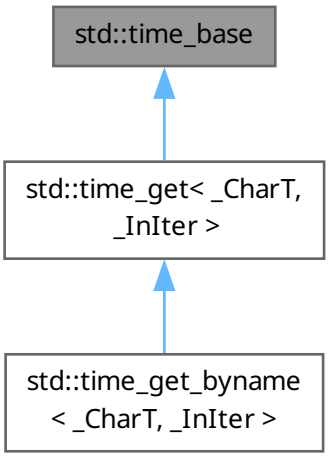
6.956.1 Detailed Description

Type throwing via random condition.
The documentation for this struct was generated from the following file:

- throw_allocator.h

6.957 std::time_base Class Reference

`#include <locale_facets_nonio.h>`
Inheritance diagram for std::time_base:



Public Types

- enum **dateorder** {
  no_order , **dmy** , **mdy** , **ymd** ,
  ydm }

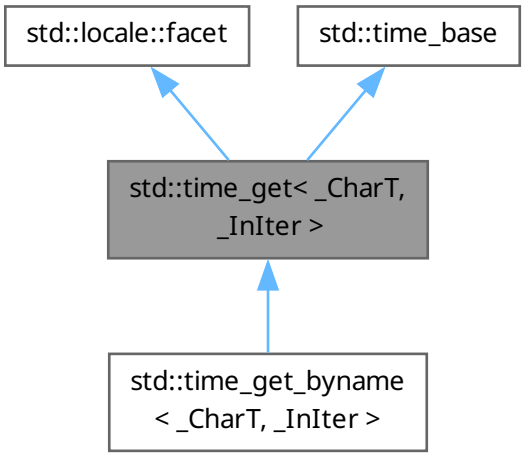
6.957.1 Detailed Description

Time format ordering data.
This class provides an enum representing different orderings of time: day, month, and year.
The documentation for this class was generated from the following file:

- locale_facets_nonio.h

6.958 std::time_get< _CharT, _InIter > Class Template Reference

`#include <locale_facets_nonio.h>`
Inheritance diagram for std::time_get< _CharT, _InIter >:



Public Types

- enum **dateorder** {
  no_order , **dmy** , **mdy** , **ymd** ,
  ydm }

- typedef _CharT char_type
- typedef _InIter iter_type

Public Member Functions

- time_get (size_t __refs=0)
- dateorder date_order () const

- iter_type get (iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, char __format, char __modifier=0) const
- iter_type get (iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const char_type *__fmt, const char_type *__fmtend) const
- iter_type get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_monthname (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_weekday (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~time_get ()
- iter_type **_M_extract_name** (iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const
- iter_type **_M_extract_num** (iter_type __beg, iter_type __end, int &__member, int __min, int __max, size_t __len, ios_base &__io, ios_base::iostate &__err) const
- iter_type **_M_extract_via_format** (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const _CharT *__format) const
- iter_type **_M_extract_via_format** (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const _CharT *__format, __time_get_state &__state) const
- iter_type **_M_extract_wday_or_month** (iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const
- virtual dateorder do_date_order () const
- iter_type do_get (iter_type __s, iter_type __end, ios_base &__f, ios_base::iostate &__err, tm *__tm, char __format, char __modifier) const
- virtual iter_type do_get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_monthname (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_weekday (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.958.1 Detailed Description

template<typename _CharT, typename _InIter>
class std::time_get< _CharT, _InIter >

Primary class template time_get.
This facet encapsulates the code to parse and return a date or time from a string. It is used by the istream numeric extraction operators.
The time_get template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the time_get facet.

6.958.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _InIter>
typedef _CharT std::time_get< _CharT, _InIter >::char_type
```

Public typedefs.

iter_type

```
template<typename _CharT, typename _InIter>
typedef _InIter std::time_get< _CharT, _InIter >::iter_type
```

Public typedefs.

6.958.3 Constructor & Destructor Documentation

time_get()

```
template<typename _CharT, typename _InIter>
std::time_get< _CharT, _InIter >::time_get (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().

~time_get()

```
template<typename _CharT, typename _InIter>
virtual std::time_get< _CharT, _InIter >::~time_get ()  [inline], [protected], [virtual]
```

Destructor.

6.958.4 Member Function Documentation

date_order()

```
template<typename _CharT, typename _InIter>
dateorder std::time_get< _CharT, _InIter >::date_order () const  [inline]
```

Return preferred order of month, day, and year.

This function returns an enum from time_base::dateorder giving the preferred ordering if the format *x* given to time_put::put() only uses month, day, and year. If the format *x* for the associated locale uses other fields, this function returns time_base::dateorder::noorder.
NOTE: The library always returns noorder at the moment.

Returns

A member of time_base::dateorder.

References do_date_order().

do_date_order()

```
template<typename _CharT, typename _InIter>
_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11 time_base::dateorder std::time_get< _CharT, _InIter >::do_date_order () const  [protected], [virtual]
```

Return preferred order of month, day, and year.
This function returns an enum from time_base::dateorder giving the preferred ordering if the format *x* given to time_put::put() only uses month, day, and year. This function is a hook for derived classes to change the value returned.

Returns

A member of time_base::dateorder.

Referenced by date_order().

do_get()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get (
            iter_type __s,
            iter_type __end,
            ios_base & __f,
            ios_base::iostate & __err,
            tm * __tm,
            char __format,
            char __modifier) const  [inline], [protected]
```

Parse input string according to format.
This function parses the string according to the provided format and optional modifier. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__s	Start of string to parse.
__end	End of string to parse.
__f	Source of the locale.

__err	Error flags to set.
__tm	Pointer to struct tm to fill in.
__format	Format specifier.
__modifier	Format modifier.

Returns

Iterator to first char not parsed.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::goodbit, std::use_facet(), and std::__ctype_abstract_base< _
Referenced by get(), and get().

do_get_date()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_date (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [protected], [virtual]
```

Parse input date string.
This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_date() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, and std::use_facet().
Referenced by get_date().

do_get_monthname()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_monthname (
             iter_type __beg,
             iter_type __end,
```

```
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [protected], [virtual]
```

Parse input month string.
This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_monthname() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, and std::use_facet().
Referenced by get_monthname().

do_get_time()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_time (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [protected], [virtual]
```

Parse input time string.
This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_time() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.

__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, and std::use_facet().
Referenced by get_time().

do_get_weekday()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_weekday (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [protected], [virtual]
```

Parse input weekday string.
This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_weekday() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, and std::use_facet().
Referenced by get_weekday().

do_get_year()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_year (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
```

```
            ios_base::iostate & __err,
            tm * __tm) const  [protected], [virtual]
```

Parse input year string.
This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_year() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::__ctype_abstract_base< _
and std::use_facet().
Referenced by get_year().

get() [1/2]

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get (
            iter_type __s,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm,
            char __format,
            char __modifier = 0) const  [inline]
```

Parse input string according to format.
This function calls time_get::do_get with the provided parameters.

See also

do_get() and get().

Parameters

__s	Start of string to parse.
__end	End of string to parse.

__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.
__format	Format specifier.
__modifier	Format modifier.

Returns

Iterator to first char not parsed.

References do_get().

get() [2/2]

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::get (
            iter_type __s,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm,
            const char_type * __fmt,
            const char_type * __fmtend) const  [inline]
```

Parse input string according to format.
This function parses the input string according to a provided format string. It does the inverse of time_put::put. The format string follows the format specified for strftime(3)/strptime(3). The actual parsing is done by time_get::do_get.

Parameters

__s	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.
__fmt	Start of the format string.
__fmtend	End of the format string.

Returns

Iterator to first char not parsed.

References std::ios_base::_M_getloc(), do_get(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::__ctype_abstract_base< _CharT >::is(), std::__ctype_abstract_base< _CharT >::narrow(), std::__ctype_abstract_base< _CharT > std::__ctype_abstract_base< _CharT >::toupper(), and std::use_facet().

get_date()

```
template<typename _CharT, typename _InIter>
```

```
iter_type std::time_get< _CharT, _InIter >::get_date (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [inline]
```

Parse input date string.
This function parses a date according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_date().
If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

References do_get_date().

get_monthname()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_monthname (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [inline]
```

Parse input month string.
This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_monthname().
Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.
If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.

__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

References do_get_monthname().

get_time()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_time (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [inline]
```

Parse input time string.
This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_time().
If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

References do_get_time().

get_weekday()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_weekday (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [inline]
```

Parse input weekday string.
This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_weekday().

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.
If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

References do_get_weekday().

get_year()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_year (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [inline]
```

Parse input year string.
This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_year().
4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.
If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

References do_get_year().

6.958.5 Member Data Documentation

id

```
template<typename _CharT, typename _InIter>
locale::id std::time_get< _CharT, _InIter >::id  [static]
```

Numpunct facet id.

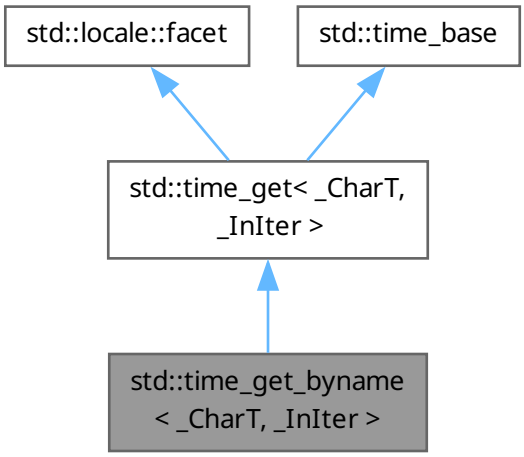
The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- locale_facets_nonio.tcc

6.959 std::time_get_byname< _CharT, _InIter > Class Template Reference

`#include <locale_facets_nonio.h>`

Inheritance diagram for std::time_get_byname< _CharT, _InIter >:



Public Types

- typedef _CharT **char_type**
- enum **dateorder** {
  no_order , **dmy** , **mdy** , **ymd** ,
  ydm }
- typedef _InIter **iter_type**

Public Member Functions

- **time_get_byname** (const char *, size_t __refs=0)
- **time_get_byname** (const string &__s, size_t __refs=0)
- dateorder date_order () const
- iter_type get (iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, char __format, char __modifier=0) const

- iter_type get (iter_type __s, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const char_type *__fmt, const char_type *__fmtend) const
- iter_type get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_monthname (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_weekday (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- iter_type get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- iter_type **_M_extract_name** (iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const
- iter_type **_M_extract_num** (iter_type __beg, iter_type __end, int &__member, int __min, int __max, size_t __len, ios_base &__io, ios_base::iostate &__err) const
- iter_type **_M_extract_via_format** (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const _CharT *__format) const
- iter_type **_M_extract_via_format** (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm, const _CharT *__format, __time_get_state &__state) const
- iter_type **_M_extract_wday_or_month** (iter_type __beg, iter_type __end, int &__member, const _CharT **__names, size_t __indexlen, ios_base &__io, ios_base::iostate &__err) const
- virtual dateorder do_date_order () const
- iter_type do_get (iter_type __s, iter_type __end, ios_base &__f, ios_base::iostate &__err, tm *__tm, char __format, char __modifier) const
- virtual iter_type do_get_date (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_monthname (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_time (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_weekday (iter_type __beg, iter_type __end, ios_base &, ios_base::iostate &__err, tm *__tm) const
- virtual iter_type do_get_year (iter_type __beg, iter_type __end, ios_base &__io, ios_base::iostate &__err, tm *__tm) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.959.1 Detailed Description

template<typename _CharT, typename _InIter>
class std::time_get_byname< _CharT, _InIter >

class time_get_byname [22.2.5.2].

6.959.2 Member Function Documentation

date_order()

```
template<typename _CharT, typename _InIter>
dateorder std::time_get< _CharT, _InIter >::date_order () const  [inline], [inherited]
```

Return preferred order of month, day, and year.
This function returns an enum from time_base::dateorder giving the preferred ordering if the format *x* given to time_put::put() only uses month, day, and year. If the format *x* for the associated locale uses other fields, this function returns time_base::dateorder::noorder.
NOTE: The library always returns noorder at the moment.

Returns

A member of time_base::dateorder.

References do_date_order().

do_date_order()

```
template<typename _CharT, typename _InIter>
_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11 time_base::dateorder std::time_get< _CharT, _InIter >::do_date_order () const  [protected], [virtual], [inherited]
```

Return preferred order of month, day, and year.
This function returns an enum from time_base::dateorder giving the preferred ordering if the format *x* given to time_put::put() only uses month, day, and year. This function is a hook for derived classes to change the value returned.

Returns

A member of time_base::dateorder.

Referenced by date_order().

do_get()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get (
            iter_type __s,
            iter_type __end,
            ios_base & __f,
            ios_base::iostate & __err,
            tm * __tm,
            char __format,
            char __modifier) const  [inline], [protected], [inherited]
```

Parse input string according to format.
This function parses the string according to the provided format and optional modifier. This function is a hook for derived classes to change the value returned.

See also

get() for more details.

Parameters

__s	Start of string to parse.

__end	End of string to parse.
__f	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.
__format	Format specifier.
__modifier	Format modifier.

Returns

Iterator to first char not parsed.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::goodbit, std::use_facet(), and std::__ctype_abstract_base< _
Referenced by get(), and get().

do_get_date()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_date (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [protected], [virtual], [inherited]
```

Parse input date string.
This function parses a date according to the format *X* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_date() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, and std::use_facet().
Referenced by get_date().

do_get_monthname()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_monthname (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [protected], [virtual], [inherited]
```

Parse input month string.
This function parses a month name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_monthname() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, and std::use_facet().
Referenced by get_monthname().

do_get_time()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_time (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [protected], [virtual], [inherited]
```

Parse input time string.
This function parses a time according to the format *x* and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_time() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, and std::use_facet().
Referenced by get_time().

do_get_weekday()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_weekday (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [protected], [virtual], [inherited]
```

Parse input weekday string.
This function parses a weekday name and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_weekday() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, and std::use_facet().
Referenced by get_weekday().

do_get_year()

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::do_get_year (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [protected], [virtual], [inherited]
```

Parse input year string.
This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. This function is a hook for derived classes to change the value returned.

See also

get_year() for details.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

References std::ios_base::_M_getloc(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::__ctype_abstract_base< _
and std::use_facet().
Referenced by get_year().

get() [1/2]

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get (
            iter_type __s,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm,
            char __format,
            char __modifier = 0) const  [inline], [inherited]
```

Parse input string according to format.
This function calls time_get::do_get with the provided parameters.

See also

do_get() and get().

Parameters

__s	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.
__format	Format specifier.
__modifier	Format modifier.

Returns

Iterator to first char not parsed.

References do_get().

get() [2/2]

```
template<typename _CharT, typename _InIter>
_InIter std::time_get< _CharT, _InIter >::get (
             iter_type __s,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm,
             const char_type * __fmt,
             const char_type * __fmtend) const  [inline], [inherited]
```

Parse input string according to format.

This function parses the input string according to a provided format string. It does the inverse of time_put::put. The format string follows the format specified for strftime(3)/strptime(3). The actual parsing is done by time_get::do_get.

Parameters

__s	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.
__fmt	Start of the format string.
__fmtend	End of the format string.

Returns

Iterator to first char not parsed.

References std::ios_base::_M_getloc(), do_get(), std::ios_base::eofbit, std::ios_base::failbit, std::ios_base::goodbit, std::__ctype_abstract_base< _CharT >::is(), std::__ctype_abstract_base< _CharT >::narrow(), std::__ctype_abstract_base< _CharT >::toupper(), and std::use_facet().

get_date()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_date (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [inline], [inherited]
```

Parse input date string.
This function parses a date according to the format *x* and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_date().
If there is a valid date string according to format *x*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the date string. If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond date string.

References do_get_date().

get_monthname()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_monthname (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [inline], [inherited]
```

Parse input month string.
This function parses a month name and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_monthname().
Parsing starts by parsing an abbreviated month name. If a valid abbreviation is followed by a character that would lead to the full month name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond month name.

References do_get_monthname().

get_time()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_time (
             iter_type __beg,
             iter_type __end,
             ios_base & __io,
             ios_base::iostate & __err,
             tm * __tm) const  [inline], [inherited]
```

Parse input time string.
This function parses a time according to the format *X* and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_time().
If there is a valid time string according to format *X*, *tm* will be filled in accordingly and the returned iterator will point to the first character beyond the time string. If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond time string.

References do_get_time().

get_weekday()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_weekday (
```

```
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [inline], [inherited]
```

Parse input weekday string.

This function parses a weekday name and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_weekday().

Parsing starts by parsing an abbreviated weekday name. If a valid abbreviation is followed by a character that would lead to the full weekday name, parsing continues until the full name is found or an error occurs. Otherwise parsing finishes at the end of the abbreviated name.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.
__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond weekday name.

References do_get_weekday().

get_year()

```
template<typename _CharT, typename _InIter>
iter_type std::time_get< _CharT, _InIter >::get_year (
            iter_type __beg,
            iter_type __end,
            ios_base & __io,
            ios_base::iostate & __err,
            tm * __tm) const  [inline], [inherited]
```

Parse input year string.

This function reads up to 4 characters to parse a year string and puts the results into a user-supplied struct tm. The result is returned by calling time_get::do_get_year().

4 consecutive digits are interpreted as a full year. If there are exactly 2 consecutive digits, the library interprets this as the number of years since 1900.

If an error occurs before the end, err |= ios_base::failbit. If parsing reads all the characters, err |= ios_base::eofbit.

Parameters

__beg	Start of string to parse.
__end	End of string to parse.
__io	Source of the locale.
__err	Error flags to set.

__tm	Pointer to struct tm to fill in.

Returns

Iterator to first char beyond year.

References do_get_year().

6.959.3 Member Data Documentation

id

```
template<typename _CharT, typename _InIter>
locale::id std::time_get< _CharT, _InIter >::id  [static], [inherited]
```

Numpunct facet id.

The documentation for this class was generated from the following file:

- locale_facets_nonio.h

6.960 std::chrono::time_point< _Clock, _Dur > Class Template Reference

Public Types

- typedef _Clock **clock**
- typedef _Dur **duration**
- typedef duration::period **period**
- typedef duration::rep **rep**

Public Member Functions

- constexpr **time_point** (const duration &__dur)
- template<typename _Dur2, typename = _Require<is_convertible<_Dur2, _Dur>>>
  constexpr **time_point** (const time_point< clock, _Dur2 > &__t)
- constexpr time_point & **operator++** ()
- constexpr time_point **operator++** (int)
- constexpr time_point & **operator+=** (const duration &__dur)
- constexpr time_point & **operator--** ()
- constexpr time_point **operator--** (int)
- constexpr time_point & **operator-=** (const duration &__dur)
- constexpr duration **time_since_epoch** () const

Static Public Member Functions

- static constexpr time_point **max** () noexcept
- static constexpr time_point **min** () noexcept

Related Symbols

(Note that these are not member symbols.)

- template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
  constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > operator+ (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

6.960.1 Detailed Description

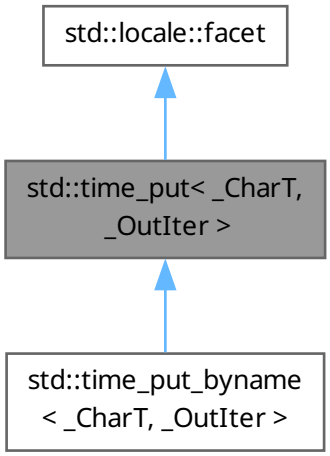
template<typename _Clock, typename _Dur>
class std::chrono::time_point< _Clock, _Dur >

`chrono::time_point` represents a point in time as measured by a clock
The documentation for this class was generated from the following file:

- chrono.h

6.961 std::time_put< _CharT, _OutIter > Class Template Reference

`#include <locale_facets_nonio.h>`
Inheritance diagram for std::time_put< _CharT, _OutIter >:



Public Types

- typedef _CharT char_type
- typedef _OutIter iter_type

Public Member Functions

- time_put (size_t __refs=0)
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, char __format, char __mod=0) const
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, const _CharT *__beg, const _CharT *__end) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual ~time_put ()
- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, char __format, char __mod) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.961.1 Detailed Description

template<typename _CharT, typename _OutIter>
class std::time_put< _CharT, _OutIter >

Primary class template time_put.
This facet encapsulates the code to format and output dates and times according to formats used by strftime().
The time_put template uses protected virtual functions to provide the actual results. The public accessors forward the call to the virtual functions. These virtual functions are hooks for developers to implement the behavior they require from the time_put facet.

6.961.2 Member Typedef Documentation

char_type

```
template<typename _CharT, typename _OutIter>
typedef _CharT std::time_put< _CharT, _OutIter >::char_type
```

Public typedefs.

iter_type

```
template<typename _CharT, typename _OutIter>
typedef _OutIter std::time_put< _CharT, _OutIter >::iter_type
```

Public typedefs.

6.961.3 Constructor & Destructor Documentation

time_put()

```
template<typename _CharT, typename _OutIter>
std::time_put< _CharT, _OutIter >::time_put (
            size_t __refs = 0)  [inline], [explicit]
```

Constructor performs initialization.
This is the constructor provided by the standard.

Parameters

__refs	Passed to the base facet class.

References std::locale::facet::facet().

~time_put()

```
template<typename _CharT, typename _OutIter>
virtual std::time_put< _CharT, _OutIter >::~time_put ()  [inline], [protected], [virtual]
```

Destructor.

6.961.4 Member Function Documentation

do_put()

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const tm * __tm,
            char __format,
            char __mod) const  [protected], [virtual]
```

Format and output a time or date.

This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

put() for more details.

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__format	Format char.
__mod	Optional modifier char.

Returns

Iterator after writing.

References std::ios_base::_M_getloc(), std::use_facet(), and std::__ctype_abstract_base< _CharT >::widen().
Referenced by put(), and put().

put() [1/2]

```
template<typename _CharT, typename _OutIter>
iter_type std::time_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const tm * __tm,
```

```
            char __format,
            char __mod = 0) const  [inline]
```

Format and output a time or date.
This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time_put::do_put().

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__format	Format char.
__mod	Optional modifier char.

Returns

Iterator after writing.

References do_put().

put() [2/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const tm * __tm,
            const _CharT * __beg,
            const _CharT * __end) const
```

Format and output a time or date.
This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__beg	Start of format string.
__end	End of format string.

Returns

Iterator after writing.

References std::ios_base::_M_getloc(), do_put(), std::__ctype_abstract_base< _CharT >::narrow(), and std::use_facet().

6.961.5 Member Data Documentation

id

```
template<typename _CharT, typename _OutIter>
locale::id std::time_put< _CharT, _OutIter >::id  [static]
```

Numpunct facet id.

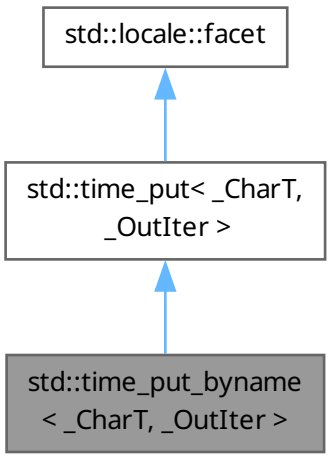
The documentation for this class was generated from the following files:

- locale_facets_nonio.h
- locale_facets_nonio.tcc

6.962 std::time_put_byname< _CharT, _OutIter > Class Template Reference

```
#include <locale_facets_nonio.h>
```

Inheritance diagram for std::time_put_byname< _CharT, _OutIter >:



Public Types

- typedef _CharT **char_type**
- typedef _OutIter **iter_type**

Public Member Functions

- **time_put_byname** (const char *, size_t __refs=0)
- **time_put_byname** (const string &__s, size_t __refs=0)
- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, char __format, char __mod=0) const

- iter_type put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, const _CharT *__beg, const _CharT *__end) const

Static Public Attributes

- static locale::id id

Protected Member Functions

- virtual iter_type do_put (iter_type __s, ios_base &__io, char_type __fill, const tm *__tm, char __format, char __mod) const

Static Protected Member Functions

- static __c_locale **_S_clone_c_locale** (__c_locale &__cloc) throw ()
- static void **_S_create_c_locale** (__c_locale &__cloc, const char *__s, __c_locale __old=0)
- static void **_S_destroy_c_locale** (__c_locale &__cloc)
- static __c_locale **_S_get_c_locale** ()
- static const char * **_S_get_c_name** () throw ()
- static __c_locale **_S_lc_ctype_c_locale** (__c_locale __cloc, const char *__s)

6.962.1 Detailed Description

template<typename _CharT, typename _OutIter>
class std::time_put_byname< _CharT, _OutIter >

class time_put_byname [22.2.5.4].

6.962.2 Member Function Documentation

do_put()

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::do_put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const tm * __tm,
            char __format,
            char __mod) const  [protected], [virtual], [inherited]
```

Format and output a time or date.
This function formats the data in struct tm according to the provided format char and optional modifier. This function is a hook for derived classes to change the value returned.

See also

put() for more details.

Parameters

__s	The stream to write to.
__io	Source of locale.

__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__format	Format char.
__mod	Optional modifier char.

Returns

Iterator after writing.

References std::ios_base::_M_getloc(), std::use_facet(), and std::__ctype_abstract_base< _CharT >::widen().
Referenced by put(), and put().

put() [1/2]

```
template<typename _CharT, typename _OutIter>
iter_type std::time_put< _CharT, _OutIter >::put (
            iter_type __s,
            ios_base & __io,
            char_type __fill,
            const tm * __tm,
            char __format,
            char __mod = 0) const  [inline], [inherited]
```

Format and output a time or date.
This function formats the data in struct tm according to the provided format char and optional modifier. The format and modifier are interpreted as by strftime(). It does so by returning time_put::do_put().

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__format	Format char.
__mod	Optional modifier char.

Returns

Iterator after writing.

References do_put().

put() [2/2]

```
template<typename _CharT, typename _OutIter>
_OutIter std::time_put< _CharT, _OutIter >::put (
            iter_type __s,
```

```
        ios_base & __io,
        char_type __fill,
        const tm * __tm,
        const _CharT * __beg,
        const _CharT * __end) const  [inherited]
```

Format and output a time or date.
This function formats the data in struct tm according to the provided format string. The format string is interpreted as by strftime().

Parameters

__s	The stream to write to.
__io	Source of locale.
__fill	char_type to use for padding.
__tm	Struct tm with date and time info to format.
__beg	Start of format string.
__end	End of format string.

Returns

Iterator after writing.

References std::ios_base::_M_getloc(), do_put(), std::__ctype_abstract_base< _CharT >::narrow(), and std::use_facet().

6.962.3 Member Data Documentation

id

```
template<typename _CharT, typename _OutIter>
locale::id std::time_put< _CharT, _OutIter >::id  [static], [inherited]
```

Numpunct facet id.
The documentation for this class was generated from the following file:

- locale_facets_nonio.h

6.963 std::timed_mutex Class Reference

```
#include <mutex>
```

Public Types

- typedef __native_type * **native_handle_type**

Public Member Functions

- **timed_mutex** (const timed_mutex &)=delete
- void **lock** ()
- native_handle_type **native_handle** () noexcept
- timed_mutex & **operator=** (const timed_mutex &)=delete
- bool **try_lock** () noexcept

- template<class _Rep, class _Period>
  bool **try_lock_for** (const chrono::duration< _Rep, _Period > &__rtime)
- template<class _Clock, class _Duration>
  bool **try_lock_until** (const chrono::time_point< _Clock, _Duration > &__atime)
- void **unlock** ()

Friends

- class **__timed_mutex_impl**< **timed_mutex** >

6.963.1 Detailed Description

The standard timed mutex type.
A non-recursive mutex that supports a timeout when trying to acquire the lock.

Since

C++11

The documentation for this class was generated from the following file:

- mutex

6.964 std::to_chars_result Struct Reference

`#include <charconv>`

Public Member Functions

- constexpr **operator bool** () const noexcept

Public Attributes

- errc **ec**
- char * **ptr**

Friends

- bool **operator==** (const to_chars_result &, const to_chars_result &)=default

6.964.1 Detailed Description

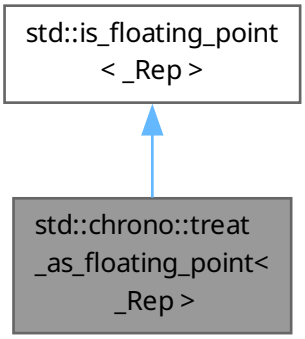
Result type of std::to_chars.
The documentation for this struct was generated from the following file:

- charconv

6.965 std::chrono::treat_as_floating_point< _Rep > Struct Template Reference

`#include <chrono.h>`

Inheritance diagram for std::chrono::treat_as_floating_point< _Rep >:



6.965.1 Detailed Description

template<typename _Rep>
struct std::chrono::treat_as_floating_point< _Rep >

Trait indicating whether to treat a type as a floating-point type.
The chrono library uses this trait to tell whether a `duration` can represent fractional values of the given precision, or only integral values.
You should specialize this trait for your own numeric types that are used with `duration` and can represent non-integral values.

Since

C++11

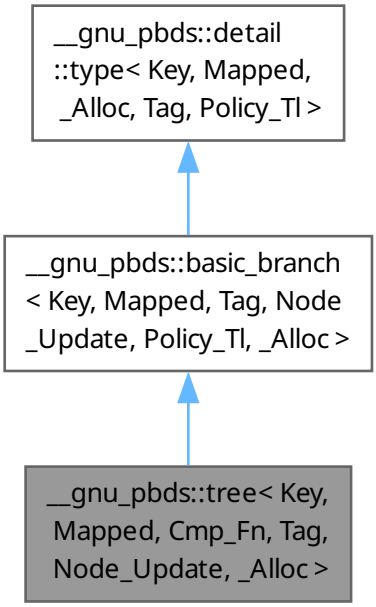
The documentation for this struct was generated from the following file:

- chrono.h

6.966 __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc > Class Template Reference

`#include <assoc_container.hpp>`

Inheritance diagram for __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >:



Public Types

- typedef Cmp_Fn cmp_fn
- typedef Node_Update **node_update**

Public Member Functions

- tree (const cmp_fn &c)
- **tree** (const tree &other)
- template<typename It>
  tree (It first, It last)
- template<typename It>
  tree (It first, It last, const cmp_fn &c)
- tree & **operator=** (const tree &other)
- void **swap** (tree &other)

6.966.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >

A tree-based container.

Template Parameters

Key	Key type.
Mapped	Map type.
Cmp_Fn	Comparison functor.
Tag	Instantiating data structure type, see container_tag.
Node_Update	Updates tree internal-nodes, restores invariants when invalidated. XXX See design::tree-based-containersnode invariants.
_Alloc	Allocator type.

Base tag choices are: ov_tree_tag, rb_tree_tag, splay_tree_tag.
Base is basic_branch.

6.966.2 Member Typedef Documentation

cmp_fn

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↩
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
typedef Cmp_Fn __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::cmp_fn
```

Comparison functor type.

6.966.3 Constructor & Destructor Documentation

tree() [1/3]

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↩
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
            const cmp_fn & c)  [inline]
```

Constructor taking some policy objects. r_cmp_fn will be copied by the Cmp_Fn object of the container object.

tree() [2/3]

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↩
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
            It first,
            It last)  [inline]
```

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

tree() [3/3]

```
template<typename Key, typename Mapped, typename Cmp_Fn = std::less<Key>, typename Tag = rb_↩
tree_tag, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ >
class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
template<typename It>
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >::tree (
            It first,
```

```
It last,
const cmp_fn & c)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects The value_types between first_it and last_it will be inserted into the container object. r_cmp_fn will be copied by the cmp_fn object of the container object.
The documentation for this class was generated from the following file:

- assoc_container.hpp

6.967 __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp > Struct Template Reference

6.967.1 Detailed Description

template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >

Tree metadata helper.
The documentation for this struct was generated from the following file:

- tree_policy/node_metadata_selector.hpp

6.968 __gnu_pbds::detail::tree_metadata_helper< Node_Update, false > Struct Template Reference

`#include <node_metadata_selector.hpp>`

Public Types

- typedef Node_Update::metadata_type **type**

6.968.1 Detailed Description

template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >

Specialization, false.
The documentation for this struct was generated from the following file:

- tree_policy/node_metadata_selector.hpp

6.969 __gnu_pbds::detail::tree_metadata_helper< Node_Update, true > Struct Template Reference

`#include <node_metadata_selector.hpp>`

Public Types

- typedef null_type **type**

6.969.1 Detailed Description

template<typename Node_Update>
struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >

Specialization, true.
The documentation for this struct was generated from the following file:

- tree_policy/node_metadata_selector.hpp

6.970 __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference

`#include <node_metadata_selector.hpp>`

Public Types

- typedef tree_metadata_helper< __node_u, null_update >::type **type**

6.970.1 Detailed Description

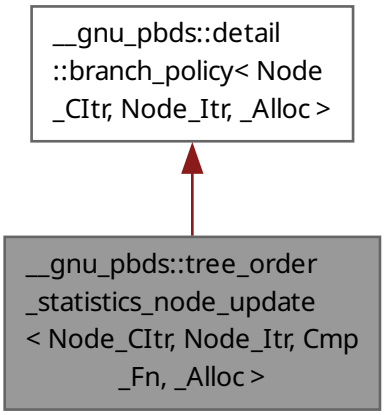
template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_CItr, typename Const_Iterator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >

Tree node metadata dispatch.
The documentation for this struct was generated from the following file:

- tree_policy/node_metadata_selector.hpp

6.971 __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc > Class Template Reference

`#include <tree_policy.hpp>`
Inheritance diagram for __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef Cmp_Fn **cmp_fn**
- typedef node_const_iterator::value_type **const_iterator**
- typedef node_iterator::value_type **iterator**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_type **key_type**

- typedef size_type **metadata_type**
- typedef Node_CItr **node_const_iterator**
- typedef Node_Itr **node_iterator**
- typedef allocator_type::size_type **size_type**

Public Member Functions

- iterator find_by_order (size_type)
- const_iterator find_by_order (size_type) const
- size_type order_of_key (key_const_reference) const

Protected Member Functions

- void operator() (node_iterator, node_const_iterator) const

6.971.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
class __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >

Functor updating ranks of entrees.

6.971.2 Member Function Documentation

find_by_order() [1/2]

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::iterator __gnu_pbds::tree_order_statist
Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
            size_type order)  [inline]
```

Finds an entry by __order. Returns an iterator to the entry with the __order order, or an iterator to the container object's end if order is at least the size of the container object.

find_by_order() [2/2]

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::const_iterator __gnu_pbds::tree_order_s
Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::find_by_order (
            size_type order) const  [inline]
```

Finds an entry by __order. Returns a const_iterator to the entry with the __order order, or a const_iterator to the container object's end if order is at least the size of the container object.

operator()()

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
void __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::↩
operator() (
            node_iterator node_it,
            node_const_iterator end_nd_it) const  [inline], [protected]
```

Updates the rank of a node through a node_iterator node_it; end_nd_it is the end node iterator.

order_of_key()

```
template<typename Node_CItr, typename Node_Itr, typename Cmp_Fn, typename _Alloc>
tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::size_type __gnu_pbds::tree_order_statis
```

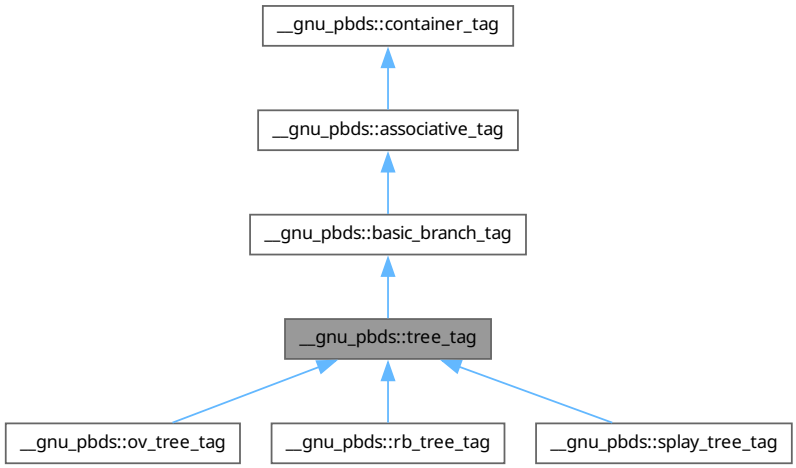
```
Node_CItr, Node_Itr, Cmp_Fn, _Alloc >::order_of_key (
            key_const_reference r_key) const  [inline]
```

Returns the order of a key within a sequence. For example, if r_key is the smallest key, this method will return 0; if r_key is a key between the smallest and next key, this method will return 1; if r_key is a key larger than the largest key, this method will return the size of r_c.
The documentation for this class was generated from the following file:

- tree_policy.hpp

6.972 __gnu_pbds::tree_tag Struct Reference

`#include <tag_and_trait.hpp>`
Inheritance diagram for __gnu_pbds::tree_tag:



6.972.1 Detailed Description

Basic tree structure.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.973 __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc > Struct Template Reference

6.973.1 Detailed Description

template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc> struct __gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >

Tree traits class, primary template.
The documentation for this struct was generated from the following file:

- branch_policy/traits.hpp

6.974 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference

`#include <traits.hpp>`

Public Types

- typedef tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type **metadata_type**
- typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc > node_const_iterator
- typedef ov_tree_node_it_< value_type, metadata_type, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node_update_pointer**

6.974.1 Detailed Description

template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >

Tree traits.

6.974.2 Member Typedef Documentation

node_const_iterator

```
template<typename Key, typename Mapped, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator
```

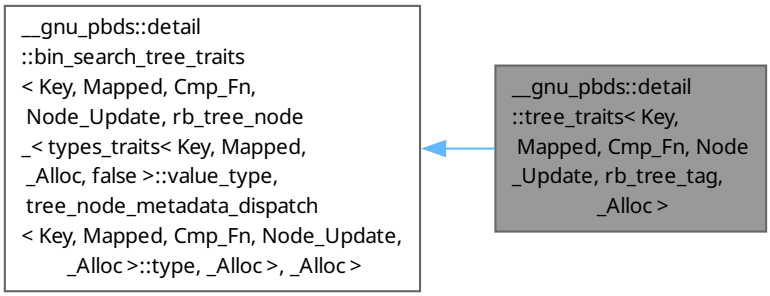
This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- ov_tree_map_/traits.hpp

6.975 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference

`#include <traits.hpp>`
Inheritance diagram for __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc

>:



Public Types

- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > **node**
- typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node_update_pointer**
- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **reverse_iterator**

6.975.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

Specialization.

6.975.2 Member Typedef Documentation

node_const_iterator

```
typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, Mapped, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _↩
Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type,
tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, _Alloc
>::node_const_iterator [inherited]
```

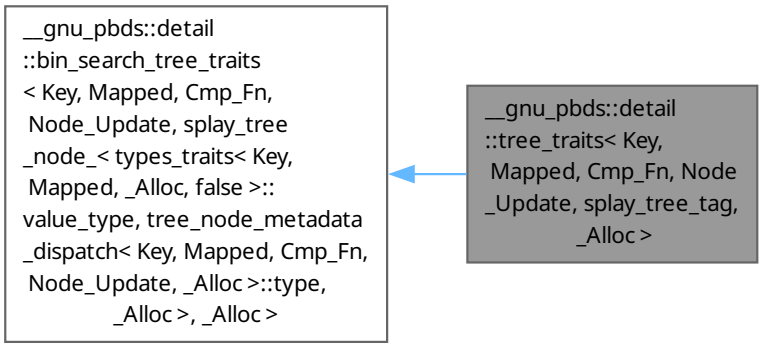
This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- rb_tree_map_/traits.hpp

6.976 __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >:



Public Types

- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > **node**
- typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _↩ Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_node_it_< splay_tree_node_< types_traits< Key, Mapped, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const↩ _iterator, point_iterator, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**

- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node_update_pointer**
- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **reverse_iterator**

6.976.1 Detailed Description

template<typename Key, typename Mapped, typename Cmp_Fn, template< typename Node_Cltr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >

Specialization.

6.976.2 Member Typedef Documentation

node_const_iterator

```
typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, Mapped, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type,
_Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, Mapped, Cmp_Fn, Node_Update, splay_tree_node_< types_traits< Key, Mapped, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, Mapped, Cmp_Fn, Node_Update, _Alloc >::type, _
Alloc >, _Alloc >::node_const_iterator [inherited]
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- splay_tree_/traits.hpp

6.977 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Public Types

- typedef tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type **metadata_type**
- typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc > node_const_iterator
- typedef node_const_iterator **node_iterator**
- typedef Node_Update< node_const_iterator, node_const_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node_update_pointer**

6.977.1 Detailed Description

template<typename Key, class Cmp_Fn, template< typename Node_Cltr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >

Specialization.

6.977.2 Member Typedef Documentation

node_const_iterator

```
template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn↩
_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef ov_tree_node_const_it_< value_type, metadata_type, _Alloc> __gnu_pbds::detail::tree_traits<
Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >::node_const_iterator
```

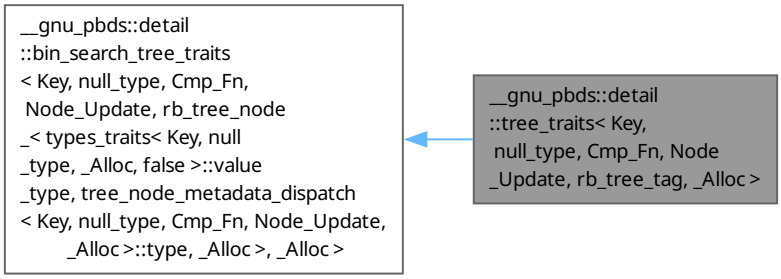
This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- ov_tree_map_/traits.hpp

6.978 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >:



Public Types

- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > **node**
- typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _↩ Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const↩ _iterator, point_iterator, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node↩ _update_pointer**
- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**

- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_↩ traits::const_reference, true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_↩ traits::const_reference, false, _Alloc > **reverse_iterator**

6.978.1 Detailed Description

template<typename Key, typename Cmp_Fn, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

Specialization.

6.978.2 Member Typedef Documentation

node_const_iterator

```
typedef bin_search_tree_const_node_it_< rb_tree_node_< types_traits< Key, null_type, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc
>::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, null_type, Cmp_Fn, Node_Update, rb_tree_node_< types_traits< Key, null_type, _Alloc, false
>::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type,
_Alloc >, _Alloc >::node_const_iterator  [inherited]
```

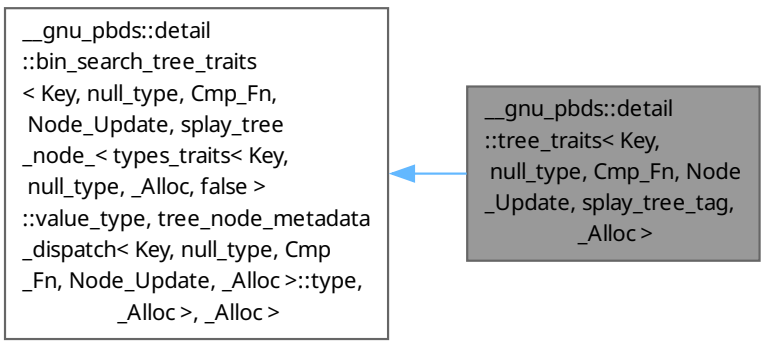
This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- rb_tree_map_/traits.hpp

6.979 __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Inheritance diagram for __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >:



Public Types

- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **const_reverse_iterator**
- typedef splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc > **node**
- typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > node_const_iterator
- typedef bin_search_tree_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc, false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc >::type, _Alloc >, point_const_iterator, point_iterator, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > **node_update**
- typedef __gnu_pbds::null_node_update< node_const_iterator, node_iterator, Cmp_Fn, _Alloc > * **null_node_update_pointer**
- typedef bin_search_tree_const_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_const_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, true, _Alloc > **point_iterator**
- typedef bin_search_tree_it_< typename node_alloc_traits::pointer, typename type_traits::value_type, typename type_traits::pointer, typename type_traits::const_pointer, typename type_traits::reference, typename type_traits::const_reference, false, _Alloc > **reverse_iterator**

6.979.1 Detailed Description

template<typename Key, class Cmp_Fn, template< typename Node_CItr, class Node_Itr, class Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >

Specialization.

6.979.2 Member Typedef Documentation

node_const_iterator

```
typedef bin_search_tree_const_node_it_< splay_tree_node_< types_traits< Key, null_type, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc
>::type, _Alloc >, point_const_iterator, point_iterator, _Alloc> __gnu_pbds::detail::bin_search_tree_traits<
Key, null_type, Cmp_Fn, Node_Update, splay_tree_node_< types_traits< Key, null_type, _Alloc,
false >::value_type, tree_node_metadata_dispatch< Key, null_type, Cmp_Fn, Node_Update, _Alloc
>::type, _Alloc >, _Alloc >::node_const_iterator  [inherited]
```

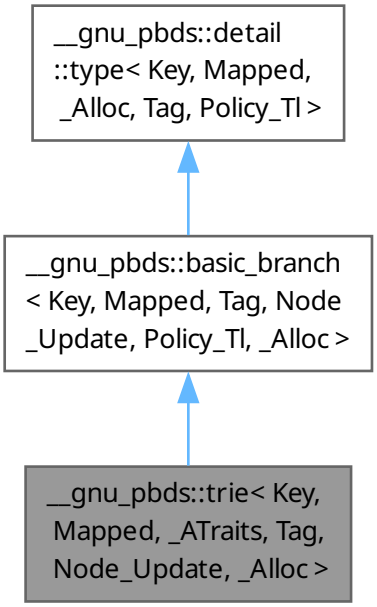
This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.
The documentation for this struct was generated from the following file:

- splay_tree_/traits.hpp

6.980 __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc > Class Template Reference

```
#include <assoc_container.hpp>
```

Inheritance diagram for __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >:



Public Types

- typedef _ATraits access_traits
- typedef Node_Update **node_update**

Public Member Functions

- trie (const access_traits &t)
- **trie** (const trie &other)
- template<typename It>
  trie (It first, It last)
- template<typename It>
  trie (It first, It last, const access_traits &t)
- trie & **operator=** (const trie &other)
- void **swap** (trie &other)

6.980.1 Detailed Description

template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc = std::allocator<char>>
class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >

A trie-based container.

Template Parameters

Key	Key type.
Mapped	Map type.
_ATraits	Element access traits.
Tag	Instantiating data structure type, see container_tag.
Node_Update	Updates trie internal-nodes, restores invariants when invalidated. XXX See design::tree-based-containersnode invariants.
_Alloc	Allocator type.

Base tag choice is pat_trie_tag.
Base is basic_branch.

6.980.2 Member Typedef Documentation

access_traits

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access↩
_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_↩
Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc
= std::allocator<char>>
typedef _ATraits __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::access_↩
traits
```

Element access traits type.

6.980.3 Constructor & Destructor Documentation

trie() [1/3]

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access↩
_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_↩
Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc
= std::allocator<char>>
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
            const access_traits & t)  [inline]
```

Constructor taking some policy objects. r_access_traits will be copied by the _ATraits object of the container object.

trie() [2/3]

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access↩
_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_↩
Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc
= std::allocator<char>>
template<typename It>
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
            It first,
            It last)  [inline]
```

Constructor taking __iterators to a range of value_types. The value_types between first_it and last_it will be inserted into the container object.

trie() [3/3]

```
template<typename Key, typename Mapped, typename _ATraits = typename detail::default_trie_access↩
_traits<Key>::type, typename Tag = pat_trie_tag, template< typename Node_CItr, typename Node_↩
```

```
Itr, typename _ATraits_, typename _Alloc_ > class Node_Update = null_node_update, typename _Alloc
= std::allocator<char>>
template<typename It>
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >::trie (
            It first,
            It last,
            const access_traits & t)  [inline]
```

Constructor taking __iterators to a range of value_types and some policy objects. The value_types between first_it and last_it will be inserted into the container object.
The documentation for this class was generated from the following file:

- assoc_container.hpp

6.981 __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp > Struct Template Reference

6.981.1 Detailed Description

template<typename Node_Update, bool _BTp>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >

Trie metadata helper.
The documentation for this struct was generated from the following file:

- trie_policy/node_metadata_selector.hpp

6.982 __gnu_pbds::detail::trie_metadata_helper< Node_Update, false > Struct Template Reference

```
#include <node_metadata_selector.hpp>
```

Public Types

- typedef Node_Update::metadata_type **type**

6.982.1 Detailed Description

template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >

Specialization, false.
The documentation for this struct was generated from the following file:

- trie_policy/node_metadata_selector.hpp

6.983 __gnu_pbds::detail::trie_metadata_helper< Node_Update, true > Struct Template Reference

```
#include <node_metadata_selector.hpp>
```

Public Types

- typedef null_type **type**

6.983.1 Detailed Description

template<typename Node_Update>
struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >

Specialization, true.
The documentation for this struct was generated from the following file:

- trie_policy/node_metadata_selector.hpp

6.984 __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc > Struct Template Reference

`#include <node_metadata_selector.hpp>`

Public Types

- typedef trie_metadata_helper< __node_u, null_update >::type **type**

6.984.1 Detailed Description

template<typename Key, typename Data, typename Cmp_Fn, template< typename Node_CItr, typename Const_Iterator, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >

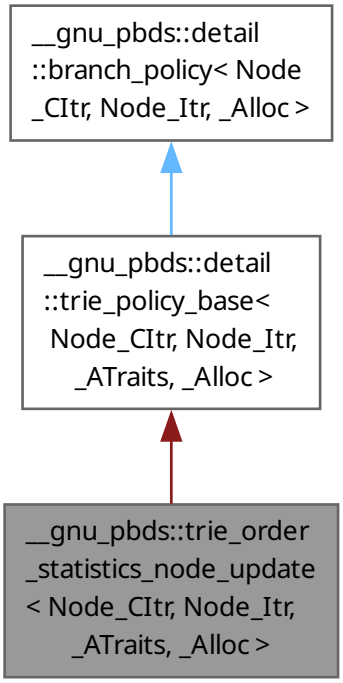
Trie node metadata dispatch.
The documentation for this struct was generated from the following file:

- trie_policy/node_metadata_selector.hpp

6.985 __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference

`#include <trie_policy.hpp>`

Inheritance diagram for __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >:



Public Types

- typedef access_traits::const_iterator **a_const_iterator**
- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef node_const_iterator::value_type **const_iterator**
- typedef node_iterator::value_type **iterator**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_type **key_type**
- typedef size_type **metadata_type**
- typedef Node_CItr **node_const_iterator**
- typedef Node_Itr **node_iterator**
- typedef allocator_type::size_type **size_type**

Public Member Functions

- iterator find_by_order (size_type)
- const_iterator find_by_order (size_type) const
- size_type order_of_key (key_const_reference) const
- size_type order_of_prefix (a_const_iterator, a_const_iterator) const

Protected Member Functions

- void operator() (node_iterator, node_const_iterator) const

Private Member Functions

- virtual const_iterator **end** () const =0
- virtual const access_traits & **get_access_traits** () const =0

6.985.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

Functor updating ranks of entrees.

6.985.2 Member Function Documentation

find_by_order() [1/2]

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::iterator __gnu_pbds::trie_order_stati
Node_CItr, Node_Itr, _ATraits, _Alloc >::find_by_order (
            size_type order)  [inline]
```

Finds an entry by __order. Returns an iterator to the entry with the __order order, or an iterator to the container object's end if order is at least the size of the container object.

find_by_order() [2/2]

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::const_iterator __gnu_pbds::trie_order
Node_CItr, Node_Itr, _ATraits, _Alloc >::find_by_order (
            size_type order) const  [inline]
```

Finds an entry by __order. Returns a const_iterator to the entry with the __order order, or a const_iterator to the container object's end if order is at least the size of the container object.

operator()()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
void __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::↩
operator() (
            node_iterator nd_it,
            node_const_iterator ) const  [inline], [protected]
```

Updates the rank of a node through a node_iterator node_it; end_nd_it is the end node iterator.

order_of_key()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::size_type __gnu_pbds::trie_order_stat
Node_CItr, Node_Itr, _ATraits, _Alloc >::order_of_key (
            key_const_reference r_key) const  [inline]
```

Returns the order of a key within a sequence. For example, if r_key is the smallest key, this method will return 0; if r_key is a key between the smallest and next key, this method will return 1; if r_key is a key larger than the largest key, this method will return the size of r_c.

order_of_prefix()

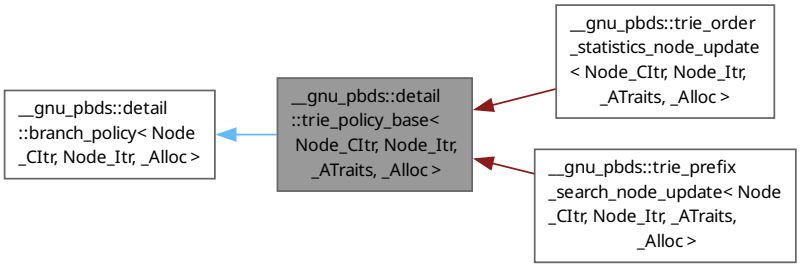
```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::size_type __gnu_pbds::trie_order_stat
Node_CItr, Node_Itr, _ATraits, _Alloc >::order_of_prefix (
            a_const_iterator b,
            a_const_iterator e) const  [inline]
```

Returns the order of a prefix within a sequence. For example, if [b, e] is the smallest prefix, this method will return 0; if r_key is a key between the smallest and next key, this method will return 1; if r_key is a key larger than the largest key, this method will return the size of r_c.
The documentation for this class was generated from the following file:

- trie_policy.hpp

6.986 __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference

`#include <trie_policy_base.hpp>`
Inheritance diagram for __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >:



Public Types

- typedef _ATraits **access_traits**
- typedef _Alloc **allocator_type**
- typedef node_const_iterator::value_type **const_iterator**
- typedef node_iterator::value_type **iterator**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_type **key_type**
- typedef null_type **metadata_type**
- typedef Node_CItr **node_const_iterator**
- typedef Node_Itr **node_iterator**
- typedef allocator_type::size_type **size_type**

Protected Types

- typedef rebind_v::const_pointer **const_pointer**
- typedef rebind_v::const_reference **const_reference**
- typedef Node_Itr::value_type **it_type**
- typedef remove_const< key_type >::type **rckey_type**
- typedef remove_const< value_type >::type **rcvalue_type**

- typedef rebind_traits< _Alloc, rckey_type > **rebind_k**
- typedef rebind_traits< _Alloc, rcvalue_type > **rebind_v**
- typedef rebind_v::reference **reference**
- typedef std::iterator_traits< it_type >::value_type **value_type**

Protected Member Functions

- virtual const_iterator **end** () const =0
- virtual iterator end ()=0
- it_type **end_iterator** () const
- virtual const access_traits & **get_access_traits** () const =0
- virtual node_const_iterator **node_begin** () const =0
- virtual node_iterator **node_begin** ()=0
- virtual node_const_iterator **node_end** () const =0
- virtual node_iterator **node_end** ()=0

Static Protected Member Functions

- static size_type **common_prefix_len** (node_iterator, e_const_iterator, e_const_iterator, const access_traits &)
- static key_const_reference **extract_key** (const_reference r_val)
- static iterator **leftmost_it** (node_iterator)
- static bool **less** (e_const_iterator, e_const_iterator, e_const_iterator, e_const_iterator, const access_traits &)
- static iterator **rightmost_it** (node_iterator)

6.986.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >

Base class for trie policies.

6.986.2 Member Function Documentation

end()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
virtual iterator __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc
>::end ()  [protected], [pure virtual]
```

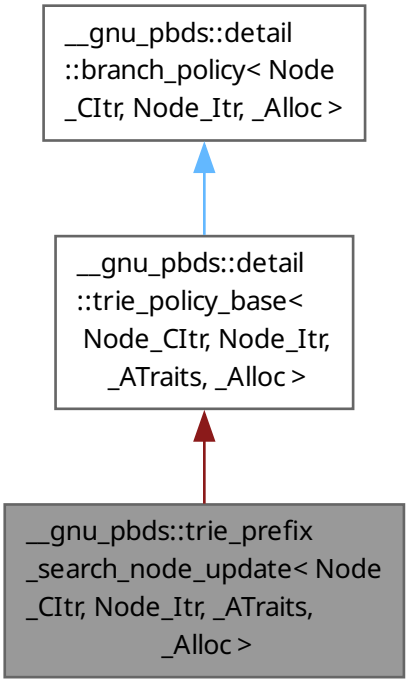
Implements __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc >.
The documentation for this class was generated from the following file:

- trie_policy_base.hpp

6.987 __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc > Class Template Reference

```
#include <trie_policy.hpp>
```

Inheritance diagram for __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >:



Public Types

- typedef access_traits::const_iterator a_const_iterator
- typedef _ATraits access_traits
- typedef _Alloc allocator_type
- typedef node_const_iterator::value_type **const_iterator**
- typedef node_iterator::value_type **iterator**
- typedef base_type::key_const_reference **key_const_reference**
- typedef base_type::key_type **key_type**
- typedef null_type **metadata_type**
- typedef Node_CItr **node_const_iterator**
- typedef Node_Itr **node_iterator**
- typedef allocator_type::size_type size_type

Public Member Functions

- std::pair< iterator, iterator > prefix_range (a_const_iterator, a_const_iterator)
- std::pair< const_iterator, const_iterator > prefix_range (a_const_iterator, a_const_iterator) const
- std::pair< iterator, iterator > prefix_range (key_const_reference)
- std::pair< const_iterator, const_iterator > prefix_range (key_const_reference) const

Protected Member Functions

- void operator() (node_iterator node_it, node_const_iterator end_nd_it) const

6.987.1 Detailed Description

template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
class __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

A node updator that allows tries to be searched for the range of values that match a certain prefix.

6.987.2 Member Typedef Documentation

a_const_iterator

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef access_traits::const_iterator __gnu_pbds::trie_prefix_search_node_update< Node_CItr,
Node_Itr, _ATraits, _Alloc >::a_const_iterator
```

Const element iterator.

access_traits

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef _ATraits __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _↩
Alloc >::access_traits
```

Element access traits.

allocator_type

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef _Alloc __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::allocator_type
```

_Alloc type.

size_type

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
typedef allocator_type::size_type __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_↩
Itr, _ATraits, _Alloc >::size_type
```

Size type.

6.987.3 Member Function Documentation

operator()()

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
void __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::↩
operator() (
            node_iterator node_it,
            node_const_iterator end_nd_it) const  [inline], [protected]
```

Called to update a node's metadata.

prefix_range() [1/4]

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::↩
iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::↩
```

```
iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::prefix_range (
              a_const_iterator b,
              a_const_iterator e)
```

Finds the iterator range corresponding to all values whose prefixes match [b, e).

prefix_range() [2/4]

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::↩
const_iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::const_iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, ↩
_Alloc >::prefix_range (
              a_const_iterator b,
              a_const_iterator e) const
```

Finds the const iterator range corresponding to all values whose prefixes match [b, e).

prefix_range() [3/4]

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::↩
iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::↩
iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::prefix_range (
              key_const_reference r_key)
```

Finds the iterator range corresponding to all values whose prefixes match r_key.

prefix_range() [4/4]

```
template<typename Node_CItr, typename Node_Itr, typename _ATraits, typename _Alloc>
std::pair< typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >::↩
const_iterator, typename trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc
>::const_iterator > __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, ↩
_Alloc >::prefix_range (
              key_const_reference r_key) const
```

Finds the const iterator range corresponding to all values whose prefixes match r_key.
The documentation for this class was generated from the following file:

- trie_policy.hpp

6.988 __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc > Struct Template Reference

```
#include <trie_policy.hpp>
```

Public Types

- enum { **reverse** }
- enum { **min_e_val** , **max_e_val** , **max_size** }
- typedef detail::__conditional_type< Reverse, typenameString::const_reverse_iterator, typenameString::const_↩ iterator >::__type const_iterator
- typedef std::iterator_traits< const_iterator >::value_type e_type
- typedef detail::rebind_traits< _Alloc, key_type >::const_reference **key_const_reference**
- typedef String **key_type**
- typedef _Alloc::size_type **size_type**

Static Public Member Functions

- static const_iterator begin (key_const_reference)
- static size_type e_pos (e_type e)
- static const_iterator end (key_const_reference)

6.988.1 Detailed Description

template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>> struct __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >

Element access traits for string types.

Template Parameters

String	String type.
Min_E_Val	Minimal element value.
Max_E_Val	Maximum element value.
Reverse	Reverse iteration should be used. Default: false.
_Alloc	Allocator type.

6.988.2 Member Typedef Documentation

const_iterator

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>>
typedef detail::__conditional_type<Reverse,typenameString::const_reverse_iterator,typename String::const_iterator>::__type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator
```

Element const iterator type.

e_type

```
template<typename String = std::string, typename String::value_type Min_E_Val = detail::__numeric_traits<typename String::value_type>::__min, typename String::value_type Max_E_Val = detail::__numeric_traits<typename String::value_type>::__max, bool Reverse = false, typename _Alloc = std::allocator<char>>
typedef std::iterator_traits<const_iterator>::value_type __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_type
```

Element type.

6.988.3 Member Function Documentation

begin()

```
template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc>
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_strin
```

```
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::begin (
            key_const_reference r_key)  [inline], [static]
```

Returns a const_iterator to the first element of key_const_reference agumnet.

e_pos()

```
template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max↩
_E_Val, bool Reverse, typename _Alloc>
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::size_type __gnu_pbds::trie_string_acc
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::e_pos (
            e_type e)  [inline], [static]
```

Maps an element to a position.

end()

```
template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max↩
_E_Val, bool Reverse, typename _Alloc>
trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::const_iterator __gnu_pbds::trie_strin
String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::end (
            key_const_reference r_key)  [inline], [static]
```

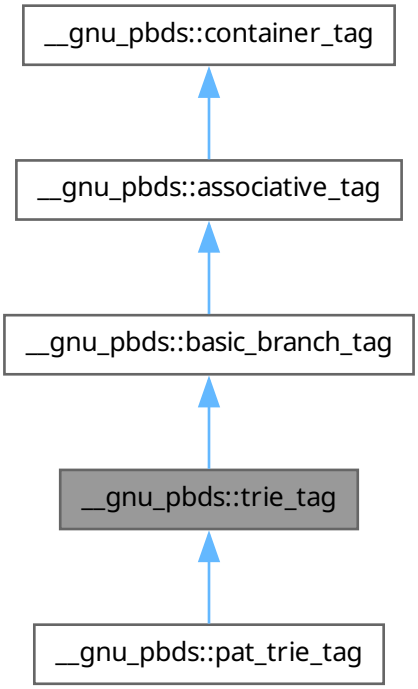
Returns a const_iterator to the after-last element of key_const_reference argument.
The documentation for this struct was generated from the following file:

- trie_policy.hpp

6.989 __gnu_pbds::trie_tag Struct Reference

```
#include <tag_and_trait.hpp>
```

Inheritance diagram for __gnu_pbds::trie_tag:



6.989.1 Detailed Description

Basic trie structure.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.990 __gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc > Struct Template Reference

6.990.1 Detailed Description

template<typename Key, typename Data, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename _ATraits_, typename _Alloc > class Node_Update, typename Tag, typename _Alloc> struct __gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >

Trie traits class, primary template.
The documentation for this struct was generated from the following file:

- branch_policy/traits.hpp

6.991 __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference

`#include <traits.hpp>`

Public Types

- typedef _ATraits **access_traits**
- typedef base_type::_CIter< node, leaf, head, inode, true > **const_iterator**
- typedef base_type::_CIter< node, leaf, head, inode, false > **const_reverse_iterator**
- typedef base_type::_Head< synth_access_traits, metadata > **head**
- typedef base_type::_Inode< synth_access_traits, metadata > **inode**
- typedef base_type::_Iter< node, leaf, head, inode, true > **iterator**
- typedef base_type::_Leaf< synth_access_traits, metadata > **leaf**
- typedef base_type::_Metadata< metadata_type, _Alloc > **metadata**
- typedef trie_node_metadata_dispatch< Key, Mapped, _ATraits, Node_Update, _Alloc >::type **metadata_type**
- typedef base_type::_Node_base< synth_access_traits, metadata > **node**
- typedef base_type::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc > node_const_iterator
- typedef base_type::_Node_iter< node, leaf, head, inode, const_iterator, iterator, _Alloc > **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc > node_update
- typedef null_node_update< node_const_iterator, node_iterator, _ATraits, _Alloc > * **null_node_update_pointer**
- typedef base_type::_Iter< node, leaf, head, inode, false > **reverse_iterator**
- typedef __gnu_pbds::detail::synth_access_traits< type_traits, false, access_traits > synth_access_traits

6.991.1 Detailed Description

template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >

Specialization.

6.991.2 Member Typedef Documentation

node_const_iterator

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef base_type::_Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::tri
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

node_update

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits<
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update
```

Type for node update.

synth_access_traits

```
template<typename Key, typename Mapped, typename _ATraits, template< typename Node_CItr, typename
Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef __gnu_pbds::detail::synth_access_traits<type_traits, false, access_traits> __gnu_pbds::detail::trie_trait
Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits
```

Type for synthesized traits.

The documentation for this struct was generated from the following file:

- pat_trie_/traits.hpp

6.992 __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc > Struct Template Reference

```
#include <traits.hpp>
```

Public Types

- typedef _ATraits **access_traits**
- typedef base_type::_CIter< node, leaf, head, inode, true > **const_iterator**
- typedef base_type::_CIter< node, leaf, head, inode, false > **const_reverse_iterator**
- typedef base_type::_Head< synth_access_traits, metadata > **head**
- typedef base_type::_Inode< synth_access_traits, metadata > **inode**
- typedef const_iterator **iterator**
- typedef base_type::_Leaf< synth_access_traits, metadata > **leaf**
- typedef base_type::_Metadata< metadata_type, _Alloc > **metadata**
- typedef trie_node_metadata_dispatch< Key, null_type, _ATraits, Node_Update, _Alloc >::type **metadata_type**
- typedef base_type::_Node_base< synth_access_traits, metadata > **node**
- typedef base_type::_Node_citer< node, leaf, head, inode, const_iterator, iterator, _Alloc > node_const_iterator
- typedef node_const_iterator **node_iterator**
- typedef Node_Update< node_const_iterator, node_iterator, _ATraits, _Alloc > node_update
- typedef null_node_update< node_const_iterator, node_const_iterator, _ATraits, _Alloc > * **null_node_update_pointer**
- typedef const_reverse_iterator **reverse_iterator**
- typedef __gnu_pbds::detail::synth_access_traits< type_traits, true, access_traits > synth_access_traits

6.992.1 Detailed Description

template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >

Specialization.

6.992.2 Member Typedef Documentation

node_const_iterator

```
template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename
Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef base_type::_Node_citer<node, leaf, head, inode, const_iterator, iterator, _Alloc> __gnu_pbds::detail::tri
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_const_iterator
```

This is an iterator to an iterator: it iterates over nodes, and de-referencing it returns one of the tree's iterators.

node_update

```
template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename
Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef Node_Update<node_const_iterator, node_iterator, _ATraits, _Alloc> __gnu_pbds::detail::trie_traits<
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::node_update
```

Type for node update.

synth_access_traits

```
template<typename Key, typename _ATraits, template< typename Node_CItr, typename Node_Itr, typename
Cmp_Fn_, typename _Alloc_ > class Node_Update, typename _Alloc>
typedef __gnu_pbds::detail::synth_access_traits<type_traits, true, access_traits> __gnu_pbds::detail::trie_traits
Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >::synth_access_traits
```

Type for synthesized traits.
The documentation for this struct was generated from the following file:

- pat_trie_/traits.hpp

6.993 __gnu_pbds::trivial_iterator_tag Struct Reference

`#include <tag_and_trait.hpp>`

6.993.1 Detailed Description

A trivial iterator tag. Signifies that the iterators has none of std::iterators's movement abilities.
The documentation for this struct was generated from the following file:

- tag_and_trait.hpp

6.994 std::try_to_lock_t Struct Reference

`#include <std_mutex.h>`

6.994.1 Detailed Description

Try to acquire ownership of the mutex without blocking.
The documentation for this struct was generated from the following file:

- std_mutex.h

6.995 std::tuple< _Elements > Class Template Reference

`#include <tuple>`

Public Member Functions

- template<typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ExplicitCtor< _Valid, _UElements... > = false>
  constexpr **tuple** (_UElements &&... __elements) noexcept(__nothrow_constructible< _UElements... >())
- template<typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ImplicitCtor< _Valid, _UElements... > = true>
  constexpr **tuple** (_UElements &&... __elements) noexcept(__nothrow_constructible< _UElements... >())
- template<typename _Alloc, _ExplicitDefaultCtor< is_object< _Alloc >::value > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a)
- template<typename _Alloc, _ImplicitDefaultCtor< is_object< _Alloc >::value > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a)

- template<typename _Alloc, typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ExplicitCtor< _Valid, _UElements... > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, _UElements &&... __elements)
- template<typename _Alloc, typename... _UElements, bool _Valid = __valid_args<_UElements...>(), _ImplicitCtor< _Valid, _UElements... > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, _UElements &&... __elements)
- template<typename _Alloc, bool _NotEmpty = (sizeof...(_Elements) >= 1), _ExplicitCtor< _NotEmpty, const _Elements &... > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const __type_identity_t< _Elements > &... __elements)
- template<typename _Alloc, bool _NotEmpty = (sizeof...(_Elements) >= 1), _ImplicitCtor< _NotEmpty, const _Elements &... > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const __type_identity_t< _Elements > &... __elements)
- template<typename _Alloc>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple &__in)
- template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<const tuple<_UElements...>&>(), _ExplicitCtor< _Valid, const _UElements &... > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)
- template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<const tuple<_UElements...>&>(), _ImplicitCtor< _Valid, const _UElements &... > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple< _UElements... > &__in)
- template<typename _Alloc>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)
- template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<tuple<_UElements...>&&>(), _ExplicitCtor< _Valid, _UElements... > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)
- template<typename _Alloc, typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<tuple<_UElements...>&&>(), _ImplicitCtor< _Valid, _UElements... > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple< _UElements... > &&__in)
- template<bool _NotEmpty = (sizeof...(_Elements) >= 1), _ExplicitCtor< _NotEmpty, const _Elements &... > = false>
  constexpr **tuple** (const __type_identity_t< _Elements > &... __elements) noexcept(__nothrow_constructible< const _Elements &... >())
- template<bool _NotEmpty = (sizeof...(_Elements) >= 1), _ImplicitCtor< _NotEmpty, const _Elements &... > = true>
  constexpr **tuple** (const __type_identity_t< _Elements > &... __elements) noexcept(__nothrow_constructible< const _Elements &... >())
- constexpr **tuple** (const tuple &)=default
- template<typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<const tuple<_UElements...>&>(), _ExplicitCtor< _Valid, const _UElements &... > = false>
  constexpr **tuple** (const tuple< _UElements... > &__in) noexcept(__nothrow_constructible< const _UElements &... >())
- template<typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<const tuple<_UElements...>&>(), _ImplicitCtor< _Valid, const _UElements &... > = true>
  constexpr **tuple** (const tuple< _UElements... > &__in) noexcept(__nothrow_constructible< const _UElements &... >())
- constexpr **tuple** (tuple &&)=default
- template<typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<tuple<_UElements...>&&>(), _ExplicitCtor< _Valid, _UElements... > = false>
  constexpr **tuple** (tuple< _UElements... > &&__in) noexcept(__nothrow_constructible< _UElements... >())
- template<typename... _UElements, bool _Valid = (sizeof...(_Elements) == sizeof...(_UElements)) && !__use_other_ctor<tuple<_UElements...>&&>(), _ImplicitCtor< _Valid, _UElements... > = true>
  constexpr **tuple** (tuple< _UElements... > &&__in) noexcept(__nothrow_constructible< _UElements... >())
- constexpr tuple & **operator=** (__conditional_t< __assignable< _Elements... >(), tuple &&, __nonesuch && > __in) noexcept(__nothrow_assignable< _Elements... >())

- constexpr tuple & **operator=** (__conditional_t< __assignable< const _Elements &... >(), const tuple &, const __nonesuch & > __in) noexcept(__nothrow_assignable< const _Elements &... >())
- template<typename... _UElements>
  constexpr __enable_if_t< __assignable< const _UElements &... >(), tuple & > **operator=** (const tuple< _↩ UElements... > &__in) noexcept(__nothrow_assignable< const _UElements &... >())
- template<typename... _UElements>
  constexpr __enable_if_t< __assignable< _UElements... >(), tuple & > **operator=** (tuple< _UElements... > &&__in) noexcept(__nothrow_assignable< _UElements... >())
- constexpr void **swap** (tuple &__in) noexcept(__and_< __is_nothrow_swappable< _Elements >... >::value)

6.995.1 Detailed Description

template<typename... _Elements>
class std::tuple< _Elements >

Primary class template, tuple.
The documentation for this class was generated from the following file:

- tuple

6.996 std::tuple< _T1, _T2 > Class Template Reference

`#include <tuple>`

Public Member Functions

- template<typename _U1, typename _U2, _ExplicitCtor<!__is_alloc_arg< _U1 >(), _U1, _U2 > = false>
  constexpr **tuple** (_U1 &&__a1, _U2 &&__a2) noexcept(__nothrow_constructible< _U1, _U2 >())
- template<typename _U1, typename _U2, _ImplicitCtor<!__is_alloc_arg< _U1 >(), _U1, _U2 > = true>
  constexpr **tuple** (_U1 &&__a1, _U2 &&__a2) noexcept(__nothrow_constructible< _U1, _U2 >())
- template<typename _Alloc, _ExplicitDefaultCtor< is_object< _Alloc >::value, _T1, _T2 > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a)
- template<typename _Alloc, _ImplicitDefaultCtor< is_object< _Alloc >::value, _T1, _T2 > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a)
- template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, _U1 &&__a1, _U2 &&__a2)
- template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, _U1 &&__a1, _U2 &&__a2)
- template<typename _Alloc, bool _Dummy = true, _ExplicitCtor< _Dummy, const _T1 &, const _T2 & > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const _T1 &__a1, const _T2 &__a2)
- template<typename _Alloc, bool _Dummy = true, _ImplicitCtor< _Dummy, const _T1 &, const _T2 & > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const _T1 &__a1, const _T2 &__a2)
- template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const pair< _U1, _U2 > &__in)
- template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const pair< _U1, _U2 > &__in)
- template<typename _Alloc>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple &__in)
- template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple< _U1, _U2 > &__in)
- template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, const tuple< _U1, _U2 > &__in)
- template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, pair< _U1, _U2 > &&__in)

- template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, pair< _U1, _U2 > &&__in)
- template<typename _Alloc>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple &&__in)
- template<typename _Alloc, typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple< _U1, _U2 > &&__in)
- template<typename _Alloc, typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>
  constexpr **tuple** (allocator_arg_t __tag, const _Alloc &__a, tuple< _U1, _U2 > &&__in)
- template<bool _Dummy = true, _ExplicitCtor< _Dummy, const _T1 &, const _T2 & > = false>
  constexpr **tuple** (const _T1 &__a1, const _T2 &__a2) noexcept(__nothrow_constructible< const _T1 &, const _T2 & >())
- template<bool _Dummy = true, _ImplicitCtor< _Dummy, const _T1 &, const _T2 & > = true>
  constexpr **tuple** (const _T1 &__a1, const _T2 &__a2) noexcept(__nothrow_constructible< const _T1 &, const _T2 & >())
- template<typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>
  constexpr **tuple** (const pair< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())
- template<typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>
  constexpr **tuple** (const pair< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())
- constexpr **tuple** (const tuple &)=default
- template<typename _U1, typename _U2, _ExplicitCtor< true, const _U1 &, const _U2 & > = false>
  constexpr **tuple** (const tuple< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())
- template<typename _U1, typename _U2, _ImplicitCtor< true, const _U1 &, const _U2 & > = true>
  constexpr **tuple** (const tuple< _U1, _U2 > &__in) noexcept(__nothrow_constructible< const _U1 &, const _U2 & >())
- template<typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>
  constexpr **tuple** (pair< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())
- template<typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>
  constexpr **tuple** (pair< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())
- constexpr **tuple** (tuple &&)=default
- template<typename _U1, typename _U2, _ExplicitCtor< true, _U1, _U2 > = false>
  constexpr **tuple** (tuple< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())
- template<typename _U1, typename _U2, _ImplicitCtor< true, _U1, _U2 > = true>
  constexpr **tuple** (tuple< _U1, _U2 > &&__in) noexcept(__nothrow_constructible< _U1, _U2 >())
- constexpr tuple & **operator=** (__conditional_t< __assignable< _T1, _T2 >(), tuple &&, __nonesuch && > __in) noexcept(__nothrow_assignable< _T1, _T2 >())
- constexpr tuple & **operator=** (__conditional_t< __assignable< const _T1 &, const _T2 & >(), const tuple &, const __nonesuch & > __in) noexcept(__nothrow_assignable< const _T1 &, const _T2 & >())
- template<typename _U1, typename _U2>
  constexpr __enable_if_t< __assignable< const _U1 &, const _U2 & >(), tuple & > **operator=** (const pair< _U1, _U2 > &__in) noexcept(__nothrow_assignable< const _U1 &, const _U2 & >())
- template<typename _U1, typename _U2>
  constexpr __enable_if_t< __assignable< const _U1 &, const _U2 & >(), tuple & > **operator=** (const tuple< _U1, _U2 > &__in) noexcept(__nothrow_assignable< const _U1 &, const _U2 & >())
- template<typename _U1, typename _U2>
  constexpr __enable_if_t< __assignable< _U1, _U2 >(), tuple & > **operator=** (pair< _U1, _U2 > &&__in) noexcept(__nothrow_assignable< _U1, _U2 >())
- template<typename _U1, typename _U2>
  constexpr __enable_if_t< __assignable< _U1, _U2 >(), tuple & > **operator=** (tuple< _U1, _U2 > &&__in) noexcept(__nothrow_assignable< _U1, _U2 >())
- constexpr void **swap** (tuple &__in) noexcept(__and_< __is_nothrow_swappable< _T1 >, __is_nothrow_↩ swappable< _T2 > >::value)

6.996.1 Detailed Description

template<typename _T1, typename _T2>
class std::tuple< _T1, _T2 >

Partial specialization, 2-element tuple. Includes construction and assignment from a pair.
The documentation for this class was generated from the following file:

- tuple

6.997 std::tuple_element< __i, _Tp > Struct Template Reference

6.997.1 Detailed Description

template<size_t __i, typename _Tp>
struct std::tuple_element< __i, _Tp >

Gives the type of the ith element of a given tuple type.
The documentation for this struct was generated from the following file:

- utility.h

6.998 std::tuple_element< 0, pair< _Tp1, _Tp2 > > Struct Template Reference

`#include <stl_pair.h>`

Public Types

- typedef _Tp1 **type**

6.998.1 Detailed Description

template<class _Tp1, class _Tp2>
struct std::tuple_element< 0, pair< _Tp1, _Tp2 > >

Partial specialization for std::pair.
The documentation for this struct was generated from the following file:

- stl_pair.h

6.999 std::tuple_element< 1, pair< _Tp1, _Tp2 > > Struct Template Reference

`#include <stl_pair.h>`

Public Types

- typedef _Tp2 **type**

6.999.1 Detailed Description

template<class _Tp1, class _Tp2>
struct std::tuple_element< 1, pair< _Tp1, _Tp2 > >

Partial specialization for std::pair.
The documentation for this struct was generated from the following file:

- stl_pair.h

6.1000 std::tuple_element< __i, tuple< _Types... > > Struct Template Reference

`#include <tuple>`

Public Types

- using **type**

6.1000.1 Detailed Description

template<size_t __i, typename... _Types>
struct std::tuple_element< __i, tuple< _Types... > >

Trait to get the Ith element type from a tuple.
The documentation for this struct was generated from the following file:

- tuple

6.1001 std::tuple_element< _Ind, array< _Tp, _Nm > > Struct Template Reference

`#include <array>`

Public Types

- using **type**

6.1001.1 Detailed Description

template<size_t _Ind, typename _Tp, size_t _Nm>
struct std::tuple_element< _Ind, array< _Tp, _Nm > >

Partial specialization for std::array.
The documentation for this struct was generated from the following file:

- array

6.1002 std::tuple_size< _Tp > Struct Template Reference

Inherited by std::tuple_size< const __enable_if_has_tuple_size< _Tp > >, std::tuple_size< const volatile __enable↩ _if_has_tuple_size< _Tp > >, and std::tuple_size< volatile __enable_if_has_tuple_size< _Tp > >.

6.1002.1 Detailed Description

template<typename _Tp>
struct std::tuple_size< _Tp >

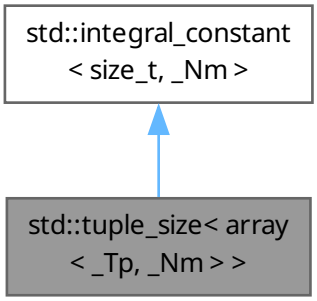
Finds the size of a given tuple type.
The documentation for this struct was generated from the following file:

- utility.h

6.1003 std::tuple_size< array< _Tp, _Nm > > Struct Template Reference

`#include <array>`

Inheritance diagram for std::tuple_size< array< _Tp, _Nm > >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr size_t **value**

6.1003.1 Detailed Description

template<**typename _Tp, size_t _Nm**>
struct std::tuple_size< **array**< **_Tp, _Nm** > >

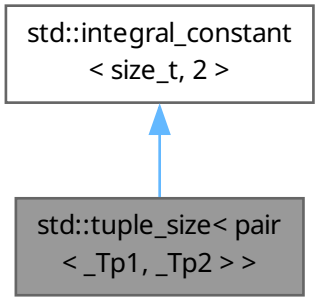
Partial specialization for std::array.
The documentation for this struct was generated from the following file:

- array

6.1004 std::tuple_size< pair< _Tp1, _Tp2 > > Struct Template Reference

`#include <stl_pair.h>`

Inheritance diagram for std::tuple_size< pair< _Tp1, _Tp2 > >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr size_t **value**

6.1004.1 Detailed Description

template<class _Tp1, class _Tp2>
struct std::tuple_size< pair< _Tp1, _Tp2 > >

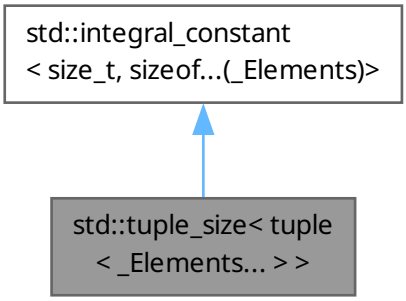
Partial specialization for std::pair.
The documentation for this struct was generated from the following file:

- stl_pair.h

6.1005 std::tuple_size< tuple< _Elements... > > Struct Template Reference

`#include <tuple>`

Inheritance diagram for std::tuple_size< tuple< _Elements... > >:



Public Types

- using **type**
- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept

Static Public Attributes

- static constexpr size_t **value**

6.1005.1 Detailed Description

template< **typename... _Elements** >
struct std::tuple_size< **tuple**< **_Elements...** > >

class tuple_size
The documentation for this struct was generated from the following file:

- tuple

6.1006 __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type Struct Reference

`#include <entry_cmp.hpp>`

Public Member Functions

- **type** (const Cmp_Fn &other)
- bool **operator()** (entry lhs, entry rhs) const

6.1006.1 Detailed Description

template<typename _VTp, typename Cmp_Fn, typename _Alloc>
struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type

Compare plus entry.
The documentation for this struct was generated from the following file:

- entry_cmp.hpp

6.1007 std::type_index Struct Reference

`#include <typeindex>`

Public Member Functions

- **type_index** (const type_info &__rhs) noexcept
- size_t **hash_code** () const noexcept
- const char * **name** () const noexcept
- bool **operator!=** (const type_index &__rhs) const noexcept
- bool **operator<** (const type_index &__rhs) const noexcept
- bool **operator<=** (const type_index &__rhs) const noexcept
- bool **operator==** (const type_index &__rhs) const noexcept
- bool **operator>** (const type_index &__rhs) const noexcept
- bool **operator>=** (const type_index &__rhs) const noexcept

6.1007.1 Detailed Description

Class type_index.
The class type_index provides a simple wrapper for type_info which can be used as an index type in associative containers (23.6) and in unordered associative containers (23.7).
The documentation for this struct was generated from the following file:

- typeindex

6.1008 __gnu_debug::type_info Class Reference

`#include <typeinfo>`

Public Member Functions

- virtual ~type_info ()
- virtual bool **__do_catch** (const type_info *__thr_type, void **__thr_obj, unsigned __outer) const
- virtual bool **__do_upcast** (const __cxxabiv1::__class_type_info *__target, void **__obj_ptr) const
- virtual bool **__is_function_p** () const
- virtual bool **__is_pointer_p** () const
- bool before (const type_info &__arg) const noexcept
- size_t **hash_code** () const noexcept
- const char * name () const noexcept
- constexpr bool **operator==** (const type_info &__arg) const noexcept

Protected Member Functions

- **type_info** (const char *__n)

Protected Attributes

- const char * **__name**

6.1008.1 Detailed Description

Part of RTTI.
The `type_info` class describes type information generated by an implementation.

6.1008.2 Constructor & Destructor Documentation

~type_info()

```
virtual std::type_info::~type_info () [virtual]
```

Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated type_info structures in the new-abi.

6.1008.3 Member Function Documentation

before()

```
bool std::type_info::before (
            const type_info & __arg) const [noexcept]
```

Returns true if `*this` precedes `__arg` in the implementation's collation order.

name()

```
const char * std::type_info::name () const [inline], [noexcept]
```

Returns an *implementation-defined* byte string; this is not portable between compilers!
The documentation for this class was generated from the following file:

- typeinfo

6.1009 std::type_info Class Reference

`#include <typeinfo>`

Public Member Functions

- virtual ~type_info ()
- virtual bool **__do_catch** (const type_info *__thr_type, void **__thr_obj, unsigned __outer) const
- virtual bool **__do_upcast** (const __cxxabiv1::__class_type_info *__target, void **__obj_ptr) const
- virtual bool **__is_function_p** () const
- virtual bool **__is_pointer_p** () const
- bool before (const type_info &__arg) const noexcept
- size_t **hash_code** () const noexcept
- const char * name () const noexcept
- constexpr bool **operator==** (const type_info &__arg) const noexcept

Protected Member Functions

- **type_info** (const char *__n)

Protected Attributes

- const char * **__name**

6.1009.1 Detailed Description

Part of RTTI.
The `type_info` class describes type information generated by an implementation.

6.1009.2 Constructor & Destructor Documentation

~type_info()

```
virtual std::type_info::~type_info ()  [virtual]
```
Destructor first. Being the first non-inline virtual function, this controls in which translation unit the vtable is emitted. The compiler makes use of that information to know where to emit the runtime-mandated type_info structures in the new-abi.

6.1009.3 Member Function Documentation

before()

```
bool std::type_info::before (
            const type_info & __arg) const  [noexcept]
```
Returns true if `*this` precedes `__arg` in the implementation's collation order.

name()

```
const char * std::type_info::name () const  [inline], [noexcept]
```
Returns an *implementation-defined* byte string; this is not portable between compilers!
The documentation for this class was generated from the following file:

- typeinfo

6.1010 std::type_order< _Tp, _Up > Struct Template Reference

`#include <compare>`

Public Types

- using **value_type**

Public Member Functions

- constexpr **operator value_type** () const noexcept
- constexpr value_type **operator()** () const noexcept

Static Public Attributes

- static constexpr strong_ordering **value**

6.1010.1 Detailed Description

template<typename _Tp, typename _Up>
struct std::type_order< _Tp, _Up >

Total ordering of types.

Since

C++26

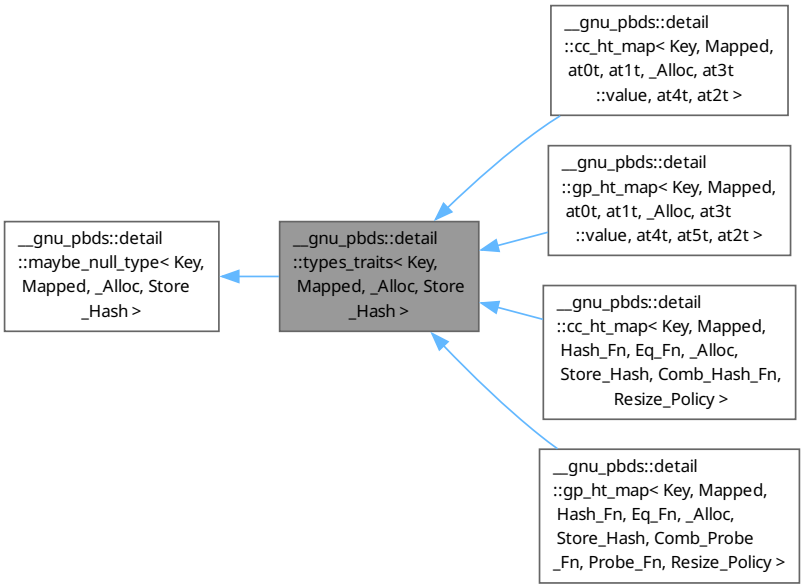
The documentation for this struct was generated from the following file:

- compare

6.1011 __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash > Struct Template Reference

`#include <types_traits.hpp>`

Inheritance diagram for __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >:



Public Types

- typedef std::pair< size_type, size_type > **comp_hash**
- typedef __rebind_va::const_pointer **const_pointer**
- typedef __rebind_va::const_reference **const_reference**
- typedef __rebind_ka::const_pointer **key_const_pointer**
- typedef __rebind_ka::const_reference **key_const_reference**
- typedef __rebind_ka::pointer **key_pointer**
- typedef __rebind_ka::reference **key_reference**
- typedef Key **key_type**
- typedef __rebind_ma::const_pointer **mapped_const_pointer**
- typedef __rebind_ma::const_reference **mapped_const_reference**
- typedef __rebind_ma::pointer **mapped_pointer**
- typedef __rebind_ma::reference **mapped_reference**
- typedef Mapped **mapped_type**
- typedef __nothrowcopy::indicator **no_throw_indicator**
- typedef __rebind_va::pointer **pointer**
- typedef __rebind_va::reference **reference**
- typedef _Alloc::size_type **size_type**
- typedef integral_constant< int, Store_Hash > **store_extra**
- typedef stored_data< value_type, size_type, Store_Hash > **stored_data_type**
- typedef select_value_type< Key, Mapped >::type **value_type**

Public Attributes

- no_throw_indicator **m_no_throw_copies_indicator**
- store_extra **m_store_extra_indicator**

6.1011.1 Detailed Description

template<typename Key, typename Mapped, typename _Alloc, bool Store_Hash>
struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >

Traits for abstract types.
The documentation for this struct was generated from the following file:

- types_traits.hpp

6.1012 std::chrono::tzdb_list Class Reference

`#include <chrono>`

Classes

- class const_iterator

Public Member Functions

- **tzdb_list** (const tzdb_list &)=delete
- const_iterator **begin** () const noexcept
- const_iterator **cbegin** () const noexcept
- const_iterator **cend** () const noexcept
- const_iterator **end** () const noexcept
- const_iterator erase_after (const_iterator __p)
- const tzdb & front () const noexcept
- tzdb_list & **operator=** (const tzdb_list &)=delete

Friends

- bool **__detail::__recent_leap_second_info** (leap_second_info &, unsigned)
- const tzdb & **get_tzdb** ()
- tzdb_list & **get_tzdb_list** ()
- class **leap_second**
- const tzdb & **reload_tzdb** ()
- struct **time_zone::_Impl**
- class **time_zone_link**
- struct **tzdb**

6.1012.1 Detailed Description

The list of `chrono::tzdb` objects
A single object of this type is constructed by the C++ runtime, and can be accessed by calling `chrono::get_tzdb_list()`.
The front of the list is the current `tzdb` object and can be accessed via `chrono::get_tzdb_list().front()` or `chrono::get_tzdb()` or `*chronoget_tzdb_list().begin()`.
The `chrono::reload_tzdb()` function will check for a newer version and if found, insert it at the front of the list.

Since

C++20

6.1012.2 Member Function Documentation

erase_after()

```
const_iterator std::chrono::tzdb_list::erase_after (
            const_iterator __p)
```

Remove the tzdb object *after* the one the iterator refers to.
Calling this function concurrently with any of `front()`, `begin()`, or `end()` does not cause a data race, but in general this function is not thread-safe. The behaviour may be undefined if erasing an element from the list while another thread is calling the same function, or incrementing an iterator into the list, or accessing the element being erased (unless it is accessed through an iterator).

Parameters

_↩ _p	A dereferenceable iterator.

Returns

An iterator the element after the one that was erased (or `end()` if there is no such element).

Since

C++20

References erase_after().
Referenced by erase_after().

front()

```
const tzdb & std::chrono::tzdb_list::front () const  [noexcept]
```

Access the current `tzdb` at the front of the list.
This returns a reference to the same object as `chrono::get_tzdb()`.

Returns

A reference to the current tzdb object.

Since

C++20

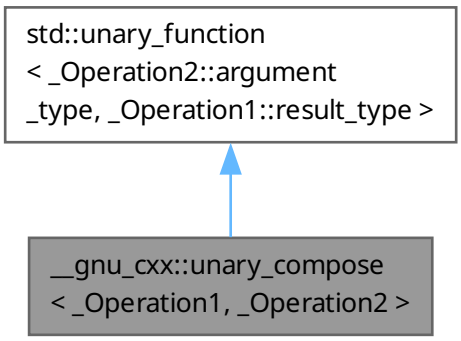
The documentation for this class was generated from the following file:

- chrono

6.1013 __gnu_cxx::unary_compose< _Operation1, _Operation2 > Class Template Reference

```
#include <functional>
```

Inheritance diagram for __gnu_cxx::unary_compose< _Operation1, _Operation2 >:



Public Types

- typedef _Operation2::argument_type argument_type
- typedef _Operation1::result_type result_type

Public Member Functions

- **unary_compose** (const _Operation1 &__x, const _Operation2 &__y)
- _Operation1::result_type **operator()** (const typename _Operation2::argument_type &__x) const

Protected Attributes

- _Operation1 **_M_fn1**
- _Operation2 **_M_fn2**

6.1013.1 Detailed Description

template<class _Operation1, class _Operation2>
class __gnu_cxx::unary_compose< _Operation1, _Operation2 >

An SGI extension .

6.1013.2 Member Typedef Documentation

argument_type

```
typedef _Operation2::argument_type std::unary_function< _Operation2::argument_type, _Operation1::↩
result_type >::argument_type  [inherited]
```

`argument_type` is the type of the argument

result_type

```
typedef _Operation1::result_type std::unary_function< _Operation2::argument_type, _Operation1::↩
result_type >::result_type  [inherited]
```

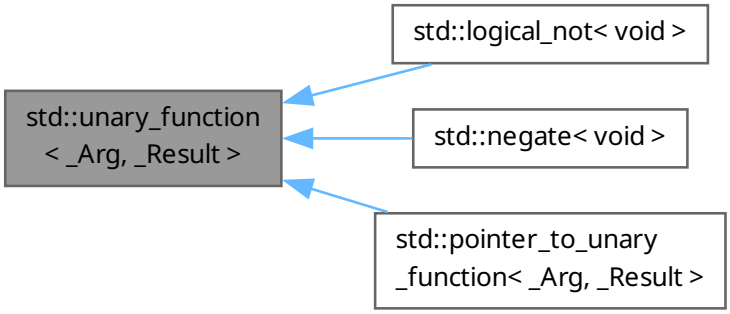
`result_type` is the return type

The documentation for this class was generated from the following file:

- ext/functional

6.1014 std::unary_function< _Arg, _Result > Struct Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::unary_function< _Arg, _Result >:



Public Types

- typedef _Arg argument_type
- typedef _Result result_type

6.1014.1 Detailed Description

template<typename _Arg, typename _Result>
struct std::unary_function< _Arg, _Result >

Helper for defining adaptable unary function objects.

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

6.1014.2 Member Typedef Documentation

argument_type

```
template<typename _Arg, typename _Result>
typedef _Arg std::unary_function< _Arg, _Result >::argument_type
```

`argument_type` is the type of the argument

result_type

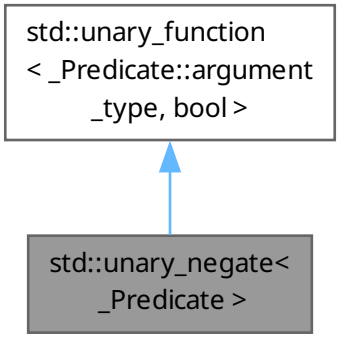
```
template<typename _Arg, typename _Result>
typedef _Result std::unary_function< _Arg, _Result >::result_type
```

`result_type` is the return type
The documentation for this struct was generated from the following file:

- stl_function.h

6.1015 std::unary_negate< _Predicate > Class Template Reference

`#include <stl_function.h>`
Inheritance diagram for std::unary_negate< _Predicate >:



Public Types

- typedef _Predicate::argument_type argument_type
- typedef bool result_type

Public Member Functions

- constexpr **unary_negate** (const _Predicate &__x)
- constexpr bool **operator()** (const typename _Predicate::argument_type &__x) const

Protected Attributes

- _Predicate **_M_pred**

6.1015.1 Detailed Description

template<**typename _Predicate**>
class std::unary_negate< **_Predicate** >

One of the negation functors.

6.1015.2 Member Typedef Documentation

argument_type

`typedef _Predicate::argument_type std::unary_function< _Predicate::argument_type, bool >::↩ argument_type [inherited]`
`argument_type` is the type of the argument

result_type

```
typedef bool std::unary_function< _Predicate::argument_type, bool >::result_type  [inherited]
```

`result_type` is the return type

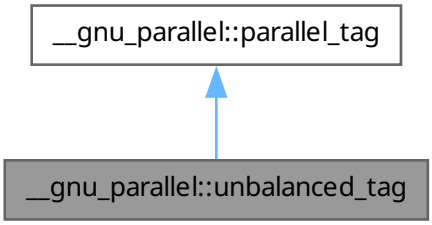
The documentation for this class was generated from the following file:

- stl_function.h

6.1016 __gnu_parallel::unbalanced_tag Struct Reference

```
#include <tags.h>
```

Inheritance diagram for __gnu_parallel::unbalanced_tag:



Public Member Functions

- _ThreadIndex __get_num_threads ()
- void set_num_threads (_ThreadIndex __num_threads)

6.1016.1 Detailed Description

Recommends parallel execution using static load-balancing at compile time.

6.1016.2 Member Function Documentation

__get_num_threads()

```
_ThreadIndex __gnu_parallel::parallel_tag::__get_num_threads ()  [inline], [inherited]
```

Find out desired number of threads.

Returns

Desired number of threads.

Referenced by __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), and __gnu_parallel::__parallel_sort(

set_num_threads()

```
void __gnu_parallel::parallel_tag::set_num_threads (
            _ThreadIndex __num_threads)  [inline], [inherited]
```

Set the desired number of threads.

Parameters

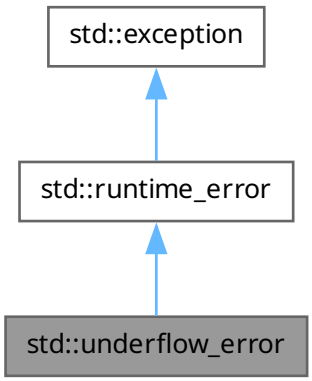
__num_threads	Desired number of threads.

The documentation for this struct was generated from the following file:

- tags.h

6.1017 std::underflow_error Class Reference

`#include <stdexcept_except.h>`
Inheritance diagram for std::underflow_error:



Public Member Functions

- **underflow_error** (const char *)
- **underflow_error** (const string &__arg)
- **underflow_error** (const underflow_error &)=default
- **underflow_error** (underflow_error &&)=default
- underflow_error & **operator=** (const underflow_error &)=default
- underflow_error & **operator=** (underflow_error &&)=default
- virtual const char * what () const noexcept

6.1017.1 Detailed Description

Thrown to indicate arithmetic underflow.

6.1017.2 Member Function Documentation

what()

`virtual const char * std::runtime_error::what () const [virtual], [noexcept], [inherited]`
Returns a C-style character string describing the general cause of the current error (the same string passed to the ctor).
Reimplemented from std::exception.
Reimplemented in std::experimental::filesystem::v1::filesystem_error, std::filesystem::filesystem_error, and std::ios_base::failure.

References what().
Referenced by what().
The documentation for this class was generated from the following file:

- stdexcept_except.h

6.1018 std::underlying_type< _Tp > Struct Template Reference

`#include <type_traits>`

6.1018.1 Detailed Description

template<typename _Tp>
struct std::underlying_type< _Tp >

The underlying type of an enum.
The documentation for this struct was generated from the following file:

- type_traits

6.1019 std::uniform_int_distribution< _IntType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _IntType result_type

Public Member Functions

- uniform_int_distribution ()
- uniform_int_distribution (_IntType __a, _IntType __b=__gnu_cxx::__int_traits< _IntType >::__max)
- **uniform_int_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomBitGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomBitGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomBitGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomBitGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomBitGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomBitGenerator &__urng, const param_type &__p)
- result_type **a** () const
- result_type **b** () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomBitGenerator>
  result_type operator() (_UniformRandomBitGenerator &__urng)
- template<typename _UniformRandomBitGenerator>
  result_type **operator()** (_UniformRandomBitGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const uniform_int_distribution &__d1, const uniform_int_distribution &__d2)

6.1019.1 Detailed Description

template<typename _IntType = int>
class std::uniform_int_distribution< _IntType >

Uniform discrete distribution for random numbers. A discrete random distribution on the range $[min, max]$ with equal probability throughout the range.

Since

C++11

6.1019.2 Member Typedef Documentation

result_type

```
template<typename _IntType = int>
typedef _IntType std::uniform_int_distribution< _IntType >::result_type
```

The type of the range of the distribution.

6.1019.3 Constructor & Destructor Documentation

uniform_int_distribution() [1/2]

```
template<typename _IntType = int>
std::uniform_int_distribution< _IntType >::uniform_int_distribution ()  [inline]
```

Constructs a uniform distribution object.

uniform_int_distribution() [2/2]

```
template<typename _IntType = int>
std::uniform_int_distribution< _IntType >::uniform_int_distribution (
            _IntType __a,
            _IntType __b = __gnu_cxx::__int_traits<_IntType>::__max)  [inline], [explicit]
```

Constructs a uniform distribution object.

6.1019.4 Member Function Documentation

max()

```
template<typename _IntType = int>
result_type std::uniform_int_distribution< _IntType >::max () const  [inline]
```

Returns the inclusive upper bound of the distribution range.

min()

```
template<typename _IntType = int>
result_type std::uniform_int_distribution< _IntType >::min () const  [inline]
```

Returns the inclusive lower bound of the distribution range.

operator()()

```
template<typename _IntType = int>
template<typename _UniformRandomBitGenerator>
result_type std::uniform_int_distribution< _IntType >::operator() (
            _UniformRandomBitGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _IntType = int>
param_type std::uniform_int_distribution< _IntType >::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
template<typename _IntType = int>
void std::uniform_int_distribution< _IntType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _IntType = int>
void std::uniform_int_distribution< _IntType >::reset ()  [inline]
```

Resets the distribution state.
Does nothing for the uniform integer distribution.

6.1019.5 Friends And Related Symbol Documentation

operator==

```
template<typename _IntType = int>
bool operator== (
            const uniform_int_distribution< _IntType > & __d1,
            const uniform_int_distribution< _IntType > & __d2)  [friend]
```

Return true if two uniform integer distributions have the same parameters.
The documentation for this class was generated from the following file:

- uniform_int_dist.h

6.1020 std::uniform_real_distribution< _RealType > Class Template Reference

```
#include <random>
```

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- uniform_real_distribution ()
- uniform_real_distribution (_RealType __a, _RealType __b=_RealType(1))
- **uniform_real_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- result_type **a** () const
- result_type **b** () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const uniform_real_distribution &__d1, const uniform_real_distribution &__d2)

6.1020.1 Detailed Description

template<typename _RealType = double>
class std::uniform_real_distribution< _RealType >

Uniform continuous distribution for random numbers.
A continuous random distribution on the range [min, max) with equal probability throughout the range. The URNG should be real-valued and deliver number in the range [0, 1).

Since

C++11

6.1020.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::uniform_real_distribution< _RealType >::result_type
```

The type of the range of the distribution.

6.1020.3 Constructor & Destructor Documentation

uniform_real_distribution() [1/2]

```
template<typename _RealType = double>
std::uniform_real_distribution< _RealType >::uniform_real_distribution ()  [inline]
```

Constructs a uniform_real_distribution object.
The lower bound is set to 0.0 and the upper bound to 1.0

uniform_real_distribution() [2/2]

```
template<typename _RealType = double>
std::uniform_real_distribution< _RealType >::uniform_real_distribution (
            _RealType __a,
            _RealType __b = _RealType(1))  [inline], [explicit]
```

Constructs a uniform_real_distribution object.

Parameters

_↩ _a	[IN] The lower bound of the distribution.
_↩ _b	[IN] The upper bound of the distribution.

6.1020.4 Member Function Documentation

max()

```
template<typename _RealType = double>
result_type std::uniform_real_distribution< _RealType >::max () const  [inline]
```

Returns the inclusive upper bound of the distribution range.

min()

```
template<typename _RealType = double>
result_type std::uniform_real_distribution< _RealType >::min () const  [inline]
```

Returns the inclusive lower bound of the distribution range.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::uniform_real_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::uniform_real_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
template<typename _RealType = double>
void std::uniform_real_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::uniform_real_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.
Does nothing for the uniform real distribution.

6.1020.5 Friends And Related Symbol Documentation

operator==

```
template<typename _RealType = double>
bool operator== (
            const uniform_real_distribution< _RealType > & __d1,
            const uniform_real_distribution< _RealType > & __d2)  [friend]
```

Return true if two uniform real distributions have the same parameters.
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.1021 std::unique_lock< _Mutex > Class Template Reference

`#include <mutex>`

Public Types

- typedef _Mutex **mutex_type**

Public Member Functions

- **unique_lock** (const unique_lock &)=delete
- **unique_lock** (mutex_type &__m)
- **unique_lock** (mutex_type &__m, adopt_lock_t) noexcept
- template<typename _Rep, typename _Period>
  unique_lock (mutex_type &__m, const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock, typename _Duration>
  unique_lock (mutex_type &__m, const chrono::time_point< _Clock, _Duration > &__atime)
- **unique_lock** (mutex_type &__m, defer_lock_t) noexcept
- **unique_lock** (mutex_type &__m, try_to_lock_t)
- **unique_lock** (unique_lock &&__u) noexcept
- void **lock** ()
- mutex_type * **mutex** () const noexcept

- **operator bool** () const noexcept
- unique_lock & **operator=** (const unique_lock &)=delete
- unique_lock & **operator=** (unique_lock &&__u) noexcept
- bool **owns_lock** () const noexcept
- mutex_type * **release** () noexcept
- void **swap** (unique_lock &__u) noexcept
- bool **try_lock** ()
- template<typename _Rep, typename _Period>
  bool **try_lock_for** (const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock, typename _Duration>
  bool **try_lock_until** (const chrono::time_point< _Clock, _Duration > &__atime)
- void **unlock** ()

Related Symbols

(Note that these are not member symbols.)

- template<typename _Mutex>
  void swap (unique_lock< _Mutex > &__x, unique_lock< _Mutex > &__y) noexcept

6.1021.1 Detailed Description

template<typename _Mutex>
class std::unique_lock< _Mutex >

A movable scoped lock type.
A unique_lock controls mutex ownership within a scope. Ownership of the mutex can be delayed until after construction and can be transferred to another unique_lock by move construction or move assignment. If a mutex lock is owned when the destructor runs ownership will be released.

Since

C++11

6.1021.2 Friends And Related Symbol Documentation

swap()

```
template<typename _Mutex>
void swap (
            unique_lock< _Mutex > & __x,
            unique_lock< _Mutex > & __y)  [related]
```

Swap overload for unique_lock objects.
The documentation for this class was generated from the following file:

- unique_lock.h

6.1022 std::unique_ptr< _Tp, _Dp > Class Template Reference

```
#include <memory>
```

Public Types

- using **deleter_type**
- using **element_type**
- using **pointer**

Public Member Functions

- template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
  constexpr unique_ptr () noexcept
- template<typename _Up, typename = _Require<is_convertible<_Up*, pointer>, is_same<_Dp, default_delete<_Tp>>>>
  unique_ptr (auto_ptr< _Up > &&__u) noexcept
- **unique_ptr** (const unique_ptr &)=delete
- template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
  constexpr unique_ptr (nullptr_t) noexcept
- template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
  constexpr unique_ptr (pointer __p) noexcept
- template<typename _Del = deleter_type, typename = _Require<is_move_constructible<_Del>>>
  constexpr unique_ptr (pointer __p, __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d) noexcept
- template<typename _Del = deleter_type, typename = _Require<is_copy_constructible<_Del>>>
  constexpr unique_ptr (pointer __p, const deleter_type &__d) noexcept
- template<typename _Del = deleter_type, typename _DelUnref = typename remove_reference<_Del>::type>
  constexpr **unique_ptr** (pointer, __enable_if_t< is_lvalue_reference< _Del >::value, _DelUnref && >)=delete
- unique_ptr (unique_ptr &&)=default
- template<typename _Up, typename _Ep, typename = _Require< __safe_conversion_up<_Up, _Ep>, __conditional_t<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>>>
  constexpr unique_ptr (unique_ptr< _Up, _Ep > &&__u) noexcept
- constexpr ~unique_ptr () noexcept
- constexpr pointer get () const noexcept
- constexpr const deleter_type & get_deleter () const noexcept
- constexpr deleter_type & get_deleter () noexcept
- constexpr operator bool () const noexcept
- constexpr add_lvalue_reference< element_type >::type operator* () const noexcept(noexcept(*std::declval< pointer >()))
- constexpr pointer operator-> () const noexcept
- unique_ptr & **operator=** (const unique_ptr &)=delete
- constexpr unique_ptr & operator= (nullptr_t) noexcept
- unique_ptr & operator= (unique_ptr &&)=default
- template<typename _Up, typename _Ep>
  constexpr enable_if< __and_< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && > >::value, unique_ptr & >::type operator= (unique_ptr< _Up, _Ep > &&__u) noexcept
- constexpr pointer release () noexcept
- constexpr void reset (pointer __p=pointer()) noexcept
- constexpr void swap (unique_ptr &__u) noexcept

Friends

- template<typename, typename, typename...>
  class **inout_ptr_t**
- template<typename, typename, typename...>
  class **out_ptr_t**

Related Symbols

(Note that these are not member symbols.)

- template<typename _CharT, typename _Traits, typename _Tp, typename _Dp>
  basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const unique_ptr< _Tp, _Dp > &__p)

- template<typename _Tp, typename _Dp>
  constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept

6.1022.1 Detailed Description

template<typename _Tp, typename _Dp = default_delete<_Tp>>
class std::unique_ptr< _Tp, _Dp >

A move-only smart pointer that manages unique ownership of a resource.

Since

C++11

6.1022.2 Constructor & Destructor Documentation

unique_ptr() [1/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp, _Dp >::unique_ptr ()  [inline], [constexpr], [noexcept]
```

Default constructor, creates a unique_ptr that owns nothing.

unique_ptr() [2/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            pointer __p)  [inline], [explicit], [constexpr], [noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an object of `element_type`

The deleter will be value-initialized.

unique_ptr() [3/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = deleter_type, typename = _Require<is_copy_constructible<_Del>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            pointer __p,
            const deleter_type & __d)  [inline], [constexpr], [noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an object of `element_type`
_↩ _d	A reference to a deleter.

The deleter will be initialized with __d

unique_ptr() [4/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = deleter_type, typename = _Require<is_move_constructible<_Del>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            pointer __p,
            __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d)  [inline], [constexpr],
[noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an object of element_type
_↩ _d	An rvalue reference to a (non-reference) deleter.

The deleter will be initialized with std::move(__d)

unique_ptr() [5/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            nullptr_t )  [inline], [constexpr], [noexcept]
```

Creates a unique_ptr that owns nothing.

unique_ptr() [6/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            unique_ptr< _Tp, _Dp > && )  [default]
```

Move constructor.

unique_ptr() [7/8]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up, typename _Ep, typename = _Require< __safe_conversion_up<_Up, _Ep>, __↩
conditional_t<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>>>>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            unique_ptr< _Up, _Ep > && __u)  [inline], [constexpr], [noexcept]
```

Converting constructor from another type.
Requires that the pointer owned by __u is convertible to the type of pointer owned by this object, __u does not own an array, and __u has a compatible deleter type.

unique_ptr() [8/8]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename>
std::unique_ptr< _Tp, _Dp >::unique_ptr (
            auto_ptr< _Up > && __u)  [inline], [noexcept]
```

Converting constructor from auto_ptr.

~unique_ptr()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::~unique_ptr ()  [inline], [constexpr], [noexcept]
```

Destructor, invokes the deleter if the stored pointer is not null.

6.1022.3 Member Function Documentation

get()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr< _Tp, _Dp >::get () const  [inline], [constexpr], [noexcept]
```

Return the stored pointer.

Referenced by std::unique_ptr< _State >::operator bool(), std::unique_ptr< _State >::operator->(), std::operator<(), std::operator<(), std::operator<(), std::unique_ptr< _State >::operator<<(), std::operator==(), std::operator>(), and std::operator>().

get_deleter() [1/2]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
const deleter_type & std::unique_ptr< _Tp, _Dp >::get_deleter () const  [inline], [constexpr],
[noexcept]
```

Return a reference to the stored deleter.

get_deleter() [2/2]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
deleter_type & std::unique_ptr< _Tp, _Dp >::get_deleter ()  [inline], [constexpr], [noexcept]
```

Return a reference to the stored deleter.

Referenced by std::unique_ptr< _State >::~unique_ptr(), and std::unique_ptr< _State >::operator=().

operator bool()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
std::unique_ptr< _Tp, _Dp >::operator bool () const  [inline], [explicit], [constexpr], [noexcept]
```

Return `true` if the stored pointer is not null.

operator*()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
add_lvalue_reference< element_type >::type std::unique_ptr< _Tp, _Dp >::operator* () const  [inline],
[constexpr], [noexcept]
```

Dereference the stored pointer.

operator->()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr< _Tp, _Dp >::operator-> () const  [inline], [constexpr], [noexcept]
```

Return the stored pointer.

operator=() [1/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
unique_ptr & std::unique_ptr< _Tp, _Dp >::operator= (
            nullptr_t )  [inline], [constexpr], [noexcept]
```

Reset the unique_ptr to empty, invoking the deleter if necessary.

operator=() [2/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
unique_ptr & std::unique_ptr< _Tp, _Dp >::operator= (
            unique_ptr< _Tp, _Dp > && )  [default]
```

Move assignment operator.
Invokes the deleter if this object owns a pointer.

operator=() [3/3]

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
template<typename _Up, typename _Ep>
enable_if< __and_< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && >
>::value, unique_ptr & >::type std::unique_ptr< _Tp, _Dp >::operator= (
            unique_ptr< _Up, _Ep > && __u)  [inline], [constexpr], [noexcept]
```

Assignment from another type.

Parameters

_↩ _u	The object to transfer ownership from, which owns a convertible pointer to a non-array object.

Invokes the deleter if this object owns a pointer.

release()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
pointer std::unique_ptr< _Tp, _Dp >::release ()  [inline], [constexpr], [noexcept]
```

Release ownership of any stored pointer.

reset()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
void std::unique_ptr< _Tp, _Dp >::reset (
            pointer __p = pointer())  [inline], [constexpr], [noexcept]
```

Replace the stored pointer.

Parameters

_↩ _p	The new pointer to store.

The deleter will be invoked if a pointer is already owned.
Referenced by std::unique_ptr< _State >::operator=(), and std::unique_ptr< _State >::operator=().

swap()

```
template<typename _Tp, typename _Dp = default_delete<_Tp>>
void std::unique_ptr< _Tp, _Dp >::swap (
            unique_ptr< _Tp, _Dp > & __u)  [inline], [constexpr], [noexcept]
```

Exchange the pointer and deleter with another object.
The documentation for this class was generated from the following files:

- unique_ptr.h
- auto_ptr.h

6.1023 std::unique_ptr< _Tp[], _Dp > Class Template Reference

`#include <memory>`

Public Types

- template<typename _Up>
  using **__safe_conversion_raw**
- template<typename _Up, typename _Ep, typename _UPtr = unique_ptr<_Up, _Ep>, typename _UP_pointer = typename _UPtr::pointer, typename _UP_element_type = typename _UPtr::element_type>
  using **__safe_conversion_up**
- using **deleter_type**
- using **element_type**
- using **pointer**

Public Member Functions

- template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
  constexpr unique_ptr () noexcept
- template<typename _Up, typename _Vp = _Dp, typename = _DeleterConstraint<_Vp>, typename = typename enable_if< __safe_conversion_raw<_Up>::value, bool>::type>
  constexpr unique_ptr (_Up __p) noexcept
- template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<_Up>, is_move_constructible<_Del>>>
  constexpr unique_ptr (_Up __p, __enable_if_t<is_lvalue_reference< _Del >::value, _Del && > __d) noexcept
- template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<_Up>, is_copy_constructible<_Del>>>
  constexpr unique_ptr (_Up __p, const deleter_type &__d) noexcept
- template<typename _Up, typename _Del = deleter_type, typename _DelUnref = typename remove_reference<_Del>::type, typename = _Require<__safe_conversion_raw<_Up>>>
  unique_ptr (_Up, __enable_if_t< is_lvalue_reference< _Del >::value, _DelUnref && >)=delete
- **unique_ptr** (const unique_ptr &)=delete
- template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
  constexpr unique_ptr (nullptr_t) noexcept
- unique_ptr (unique_ptr &&)=default
- template<typename _Up, typename _Ep, typename = _Require< __safe_conversion_up<_Up, _Ep>, __conditional_t<is_reference<_Dp>::value, is_same<_Ep, _Dp>, is_convertible<_Ep, _Dp>>>>
  constexpr **unique_ptr** (unique_ptr< _Up, _Ep > &&__u) noexcept
- constexpr ~unique_ptr ()
- constexpr pointer get () const noexcept
- constexpr const deleter_type & get_deleter () const noexcept
- constexpr deleter_type & get_deleter () noexcept
- constexpr operator bool () const noexcept
- constexpr add_lvalue_reference< element_type >::type operator* () const noexcept(noexcept(*std::declval< pointer >()))
- constexpr pointer operator-> () const noexcept
- unique_ptr & **operator=** (const unique_ptr &)=delete
- constexpr unique_ptr & operator= (nullptr_t) noexcept
- unique_ptr & operator= (unique_ptr &&)=default
- template<typename _Up, typename _Ep>
  constexpr enable_if< __and_< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && > >::value, unique_ptr & >::type operator= (unique_ptr< _Up, _Ep > &&__u) noexcept
- constexpr std::add_lvalue_reference< element_type >::type operator[ ] (size_t __i) const

- constexpr pointer release () noexcept
- template<typename _Up, typename = _Require< __or_<is_same<_Up, pointer>, __and_<is_same<pointer, element_type*>, is_pointer<_Up>, is_convertible< typename remove_pointer<_Up>::type(*)[], element_type(*)[] > > > >>
  constexpr void reset (_Up __p) noexcept
- constexpr void **reset** (nullptr_t=nullptr) noexcept
- constexpr void swap (unique_ptr &__u) noexcept

Friends

- template<typename, typename, typename...>
  class **inout_ptr_t**
- template<typename, typename, typename...>
  class **out_ptr_t**

Related Symbols

(Note that these are not member symbols.)

- basic_ostream< _CharT, _Traits > & operator<< (basic_ostream< _CharT, _Traits > &__os, const unique_ptr< _Tp, _Dp > &__p)
- constexpr enable_if< __is_swappable< _Dp >::value >::type swap (unique_ptr< _Tp, _Dp > &__x, unique_ptr< _Tp, _Dp > &__y) noexcept

6.1023.1 Detailed Description

template<typename _Tp, typename _Dp>
class std::unique_ptr< _Tp[ ], _Dp >

A move-only smart pointer that manages unique ownership of an array.

Since

C++11

6.1023.2 Constructor & Destructor Documentation

unique_ptr() [1/6]

```
template<typename _Tp, typename _Dp>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr ()  [inline], [constexpr], [noexcept]
```

Default constructor, creates a unique_ptr that owns nothing.
References unique_ptr().
Referenced by unique_ptr(), unique_ptr(), unique_ptr(), unique_ptr(), unique_ptr(), unique_ptr(), ~unique_ptr(), operator=(), operator=(), operator=(), and swap().

unique_ptr() [2/6]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Vp = _Dp, typename = _DeleterConstraint<_Vp>, typename = typename
enable_if< __safe_conversion_raw<_Up>::value, bool>::type>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
            _Up __p)  [inline], [explicit], [constexpr], [noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an array of a type safely convertible to an array of `element_type`

The deleter will be value-initialized.
References unique_ptr().

unique_ptr() [3/6]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<↩
_Up>, is_copy_constructible<_Del>>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
            _Up __p,
            const deleter_type & __d)  [inline], [constexpr], [noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an array of a type safely convertible to an array of `element_type`
_↩ _d	A reference to a deleter.

The deleter will be initialized with `__d`
References unique_ptr().

unique_ptr() [4/6]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Del = deleter_type, typename = _Require<__safe_conversion_raw<↩
_Up>, is_move_constructible<_Del>>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
            _Up __p,
            __enable_if_t<!is_lvalue_reference< _Del >::value, _Del && > __d)  [inline], [constexpr],
[noexcept]
```

Takes ownership of a pointer.

Parameters

_↩ _p	A pointer to an array of a type safely convertible to an array of `element_type`
_↩ _d	A reference to a deleter.

The deleter will be initialized with `std::move(__d)`
References unique_ptr(), and std::move().

unique_ptr() [5/6]

```
template<typename _Tp, typename _Dp>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
```

```
            unique_ptr< _Tp[], _Dp > && )  [default]
```

Move constructor.
References unique_ptr().

unique_ptr() [6/6]

```
template<typename _Tp, typename _Dp>
template<typename _Del = _Dp, typename = _DeleterConstraint<_Del>>
std::unique_ptr< _Tp[], _Dp >::unique_ptr (
            nullptr_t )  [inline], [constexpr], [noexcept]
```

Creates a unique_ptr that owns nothing.
References unique_ptr().

~unique_ptr()

```
template<typename _Tp, typename _Dp>
std::unique_ptr< _Tp[], _Dp >::~unique_ptr ()  [inline], [constexpr]
```

Destructor, invokes the deleter if the stored pointer is not null.
References unique_ptr(), and get_deleter().

6.1023.3 Member Function Documentation

get()

```
template<typename _Tp, typename _Dp>
pointer std::unique_ptr< _Tp[], _Dp >::get () const  [inline], [constexpr], [noexcept]
```

Return the stored pointer.
References get().
Referenced by get(), operator bool(), and operator[]().

get_deleter() [1/2]

```
template<typename _Tp, typename _Dp>
const deleter_type & std::unique_ptr< _Tp[], _Dp >::get_deleter () const  [inline], [constexpr],
[noexcept]
```

Return a reference to the stored deleter.
References get_deleter().

get_deleter() [2/2]

```
template<typename _Tp, typename _Dp>
deleter_type & std::unique_ptr< _Tp[], _Dp >::get_deleter ()  [inline], [constexpr], [noexcept]
```

Return a reference to the stored deleter.
References get_deleter().
Referenced by ~unique_ptr(), get_deleter(), get_deleter(), and operator=().

operator bool()

```
template<typename _Tp, typename _Dp>
std::unique_ptr< _Tp[], _Dp >::operator bool () const  [inline], [explicit], [constexpr], [noexcept]
```

Return `true` if the stored pointer is not null.
References get().

operator*()

```
add_lvalue_reference< element_type >::type std::unique_ptr< _Tp, _Dp >::operator* () const  [inline],
[constexpr], [noexcept]
```

Dereference the stored pointer.

operator->()

```
pointer std::unique_ptr< _Tp, _Dp >::operator-> () const  [inline], [constexpr], [noexcept]
```

Return the stored pointer.

operator=() [1/3]

```
template<typename _Tp, typename _Dp>
unique_ptr & std::unique_ptr< _Tp[], _Dp >::operator= (
            nullptr_t )  [inline], [constexpr], [noexcept]
```

Reset the unique_ptr to empty, invoking the deleter if necessary.
References unique_ptr(), operator=(), and reset().

operator=() [2/3]

```
template<typename _Tp, typename _Dp>
unique_ptr & std::unique_ptr< _Tp[], _Dp >::operator= (
            unique_ptr< _Tp[], _Dp > && )  [default]
```

Move assignment operator.
Invokes the deleter if this object owns a pointer.
References unique_ptr(), and operator=().
Referenced by operator=(), operator=(), and operator=().

operator=() [3/3]

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename _Ep>
enable_if< __and_< __safe_conversion_up< _Up, _Ep >, is_assignable< deleter_type &, _Ep && >
>::value, unique_ptr & >::type std::unique_ptr< _Tp[], _Dp >::operator= (
            unique_ptr< _Up, _Ep > && __u)  [inline], [constexpr], [noexcept]
```

Assignment from another type.

Parameters

_↩ _u	The object to transfer ownership from, which owns a convertible pointer to an array object.

Invokes the deleter if this object owns a pointer.
References unique_ptr(), std::forward(), get_deleter(), operator=(), and reset().

operator[]()

```
template<typename _Tp, typename _Dp>
std::add_lvalue_reference< element_type >::type std::unique_ptr< _Tp[], _Dp >::operator[] (
            size_t __i) const  [inline], [constexpr]
```

Access an element of owned array.
References get(), and operator[]().
Referenced by operator[]().

release()

```
template<typename _Tp, typename _Dp>
pointer std::unique_ptr< _Tp[], _Dp >::release ()  [inline], [constexpr], [noexcept]
```

Release ownership of any stored pointer.
References release().
Referenced by release().

reset()

```
template<typename _Tp, typename _Dp>
template<typename _Up, typename = _Require< __or_<is_same<_Up, pointer>, __and_<is_same<pointer,
element_type*>, is_pointer<_Up>, is_convertible< typename remove_pointer<_Up>::type(*)[], element↩
_type(*)[] > > > >>
void std::unique_ptr< _Tp[], _Dp >::reset (
            _Up __p)  [inline], [constexpr], [noexcept]
```

Replace the stored pointer.

Parameters

_↩ _p	The new pointer to store.

The deleter will be invoked if a pointer is already owned.
References std::move(), and reset().
Referenced by operator=(), operator=(), and reset().

swap()

```
template<typename _Tp, typename _Dp>
void std::unique_ptr< _Tp[], _Dp >::swap (
            unique_ptr< _Tp[], _Dp > & __u)  [inline], [constexpr], [noexcept]
```

Exchange the pointer and deleter with another object.
References unique_ptr(), and swap().
Referenced by swap().

6.1023.4 Friends And Related Symbol Documentation

operator<<()

```
basic_ostream< _CharT, _Traits > & operator<< (
            basic_ostream< _CharT, _Traits > & __os,
            const unique_ptr< _Tp, _Dp > & __p)  [related]
```

Stream output operator for unique_ptr.

Since

C++20

swap()

```
enable_if< __is_swappable< _Dp >::value >::type swap (
            unique_ptr< _Tp, _Dp > & __x,
            unique_ptr< _Tp, _Dp > & __y)  [related]
```

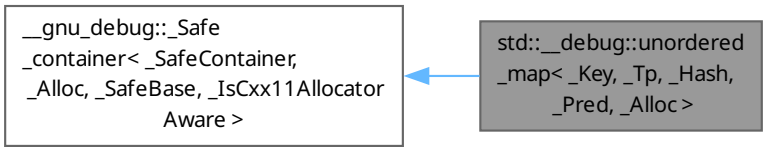
Swap overload for unique_ptr.
The documentation for this class was generated from the following files:

- unique_ptr.h
- auto_ptr.h

6.1024 std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_map>`

Inheritance diagram for std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_map > **const_iterator**
- typedef __gnu_debug::_Safe_local_iterator< _Base_const_local_iterator, unordered_map > **const_local_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, unordered_map > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef __gnu_debug::_Safe_local_iterator< _Base_local_iterator, unordered_map > **local_iterator**
- typedef _Base::mapped_type **mapped_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_map** (_Base_ref __x)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())

- **unordered_map** (const allocator_type &__a)
- **unordered_map** (const unordered_map &)=default
- **unordered_map** (const unordered_map &__umap, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** (size_type __n, const allocator_type &__a)
- **unordered_map** (size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_map** (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** (unordered_map &&)=default
- **unordered_map** (unordered_map &&__umap, const allocator_type &__a) noexcept(noexcept(_Base(std::move(__umap), __a)))
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- local_iterator **begin** (size_type __b)
- const_local_iterator **begin** (size_type __b) const
- size_type **bucket_size** (size_type __b) const
- const_iterator **cbegin** () const noexcept
- const_local_iterator **cbegin** (size_type __b) const
- const_iterator **cend** () const noexcept
- const_local_iterator **cend** (size_type __b) const
- void **clear** () noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __hint, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- local_iterator **end** (size_type __b)
- const_local_iterator **end** (size_type __b) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- _Base_iterator **erase** (_Base_const_iterator __it)
- template<__heterogeneous_hash_key< unordered_map > _Kt>
  size_type **erase** (_Kt &&__key)
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __it)
- iterator **erase** (iterator __it)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  std::pair< iterator, bool > **insert** (_Pair &&__obj)

- std::pair< iterator, bool > **insert** (const value_type &__obj)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type> iterator **insert** (const_iterator __hint, _Pair &&__obj)
- iterator **insert** (const_iterator __hint, const value_type &__obj)
- iterator **insert** (const_iterator __hint, value_type &&__x)
- void **insert** (std::initializer_list< value_type > __l)
- std::pair< iterator, bool > **insert** (value_type &&__x)
- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __f)
- unordered_map & **operator=** (const unordered_map &)=default
- unordered_map & **operator=** (initializer_list< value_type > __l)
- unordered_map & **operator=** (unordered_map &&)=default
- void **swap** (unordered_map &__x) noexcept(noexcept(declval< _Base & >().swap(__x)))

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT> class **::__gnu_debug::_Safe_iterator**
- template<typename _ItT, typename _SeqT> class **::__gnu_debug::_Safe_local_iterator**

6.1024.1 Detailed Description

template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> > >
class std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >

Class std::unordered_map with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/unordered_map

6.1025 std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_map>`

Public Types

- typedef _Hashtable::key_type key_type
- typedef _Hashtable::value_type value_type
- typedef _Hashtable::mapped_type mapped_type
- typedef _Hashtable::hasher hasher
- typedef _Hashtable::key_equal key_equal
- typedef _Hashtable::allocator_type allocator_type

- typedef _Hashtable::pointer pointer
- typedef _Hashtable::const_pointer const_pointer
- typedef _Hashtable::reference reference

- typedef _Hashtable::const_reference const_reference
- typedef _Hashtable::iterator iterator
- typedef _Hashtable::const_iterator const_iterator
- typedef _Hashtable::local_iterator local_iterator
- typedef _Hashtable::const_local_iterator const_local_iterator
- typedef _Hashtable::size_type size_type
- typedef _Hashtable::difference_type difference_type

Public Member Functions

- unordered_map ()=default
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_map (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_map (const allocator_type &__a)
- unordered_map (const unordered_map &)=default
- **unordered_map** (const unordered_map &__umap, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_map** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_map (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_map** (size_type __n, const allocator_type &__a)
- **unordered_map** (size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_map (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_map (unordered_map &&)=default
- **unordered_map** (unordered_map &&__umap, const allocator_type &__a) noexcept(noexcept(_Hashtable(std::move(__umap._M_h), __a)))
- iterator begin () noexcept
- local_iterator begin (size_type __n)
- size_type bucket_count () const noexcept
- size_type **bucket_size** (size_type __n) const
- void clear () noexcept
- template<typename... _Args>
  std::pair< iterator, bool > emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- iterator end () noexcept
- local_iterator end (size_type __n)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- hasher hash_function () const

- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_equal key_eq () const
- float load_factor () const noexcept
- size_type max_bucket_count () const noexcept
- float max_load_factor () const noexcept
- void max_load_factor (float __z)
- size_type max_size () const noexcept
- unordered_map & operator= (const unordered_map &)=default
- unordered_map & operator= (initializer_list< value_type > __l)
- unordered_map & operator= (unordered_map &&)=default
- void rehash (size_type __n)
- void reserve (size_type __n)
- size_type size () const noexcept
- void swap (unordered_map &__x) noexcept(noexcept(_M_h.swap(__x._M_h)))

- const_iterator begin () const noexcept
- const_iterator cbegin () const noexcept

- const_iterator end () const noexcept
- const_iterator cend () const noexcept

- std::pair< iterator, bool > insert (const value_type &__x)
- std::pair< iterator, bool > insert (value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair && >::value, pair< iterator, bool > > insert (_Pair &&__x)

- iterator insert (const_iterator __hint, const value_type &__x)
- iterator insert (const_iterator __hint, value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > insert (const_iterator __hint, _Pair &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __position)

- size_type erase (const key_type &__x)
- template<__heterogeneous_hash_key< unordered_map > _Kt>
  size_type erase (_Kt &&__x)

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const

- size_type count (const key_type &__x) const

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_h._M_find_tr(__x), void(), true)

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const

- mapped_type & operator[ ] (const key_type &__k)
- mapped_type & operator[ ] (key_type &&__k)
- template<__heterogeneous_hash_key< unordered_map > _Kt>
  mapped_type & operator[ ] (_Kt &&__k)

- mapped_type & at (const key_type &__k)
- template<__heterogeneous_hash_key< unordered_map > _Kt>
  mapped_type & at (const _Kt &__k)
- const mapped_type & at (const key_type &__k) const
- template<__heterogeneous_hash_key< unordered_map > _Kt>
  const mapped_type & at (const _Kt &__k) const

- size_type **bucket** (const key_type &__key) const
- template<__heterogeneous_hash_key< unordered_map > _Kt>
  size_type **bucket** (const _Kt &__key) const

- const_local_iterator begin (size_type __n) const
- const_local_iterator cbegin (size_type __n) const

- const_local_iterator end (size_type __n) const
- const_local_iterator cend (size_type __n) const

Friends

- template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1>
  bool **operator==** (const unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &, const unordered_map< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &)

6.1025.1 Detailed Description

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >

A standard container composed of unique keys (containing at most one of each key value) that associates values of another type with the keys.

Since

C++11

Template Parameters

_Key	Type of key objects.
_Tp	Type of mapped objects.
_Hash	Hashing function object type, defaults to hash<_Key>.
_Pred	Predicate function object type, defaults to equal_to<_Key>.
_Alloc	Allocator type, defaults to std::allocator<std::pair<const _Key, _Tp>>.

Meets the requirements of a container, and unordered associative container
The resulting value type of the container is std::pair<const _Key, _Tp>.
Base is _Hashtable, dispatched at compile time via template alias __umap_hashtable.

6.1025.2 Member Typedef Documentation

allocator_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::allocator_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::allocator_type
```

Public typedefs.

const_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_iterator
```

Iterator-related typedefs.

const_local_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_local_iterator
```

Iterator-related typedefs.

const_pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
typedef _Hashtable::const_pointer std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::const_↩
pointer
```
Iterator-related typedefs.

const_reference

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_reference std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::↩
const_reference
```
Iterator-related typedefs.

difference_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::difference_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::↩
difference_type
```
Iterator-related typedefs.

hasher

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::hasher std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::hasher
```
Public typedefs.

iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::iterator
```
Iterator-related typedefs.

key_equal

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_equal std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_equal
```
Public typedefs.

key_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type
```
Public typedefs.

local_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::local↩
_iterator
```
Iterator-related typedefs.

mapped_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::mapped_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::mapped_↩
type
```

Public typedefs.

pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::pointer std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

reference

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::reference std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

size_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::size_type
```

Iterator-related typedefs.

value_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::value_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::value_type
```

Public typedefs.

6.1025.3 Constructor & Destructor Documentation

unordered_map() [1/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map ()  [default]
```

Default constructor.

unordered_map() [2/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
            size_type __n,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Default constructor creates no elements.

Parameters

__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

unordered_map() [3/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
            _InputIterator __first,
            _InputIterator __last,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_map from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_map consisting of copies of the elements from [__first,__last). This is linear in N (where N is distance(__first,__last)).

unordered_map() [4/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
            const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & )  [default]
```

Copy constructor.

unordered_map() [5/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
```

```
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
            unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > && )  [default]
```

Move constructor.

unordered_map() [6/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
            const allocator_type & __a)  [inline], [explicit]
```

Creates an unordered_map with no elements.

Parameters

_↩ _a	An allocator object.

unordered_map() [7/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_map (
            initializer_list< value_type > __l,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_map from an initializer_list.

Parameters

__l	An initializer_list.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_map consisting of copies of the elements in the list. This is linear in N (where N is *__l.size()*).

6.1025.4 Member Function Documentation

at() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
```

```
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<__heterogeneous_hash_key< unordered_map > _Kt>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (
            const _Kt & __k)  [inline]
```

Access to unordered_map data.

Parameters

_↩ _k	The key for which data should be retrieved.

Returns

A reference to the data whose key is equal to __k, if such a data is present in the unordered_map.

Exceptions

std::out_of_range	If no such data is present.

at() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<__heterogeneous_hash_key< unordered_map > _Kt>
const mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (
            const _Kt & __k) const  [inline]
```

Access to unordered_map data.

Parameters

_↩ _k	The key for which data should be retrieved.

Returns

A reference to the data whose key is equal to __k, if such a data is present in the unordered_map.

Exceptions

std::out_of_range	If no such data is present.

at() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (
            const key_type & __k)  [inline]
```

Access to unordered_map data.

Parameters

_↩ _k	The key for which data should be retrieved.

Returns

A reference to the data whose key is equal to __k, if such a data is present in the unordered_map.

Exceptions

std::out_of_range	If no such data is present.

at() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::at (
            const key_type & __k) const  [inline]
```

Access to unordered_map data.

Parameters

_↩ _k	The key for which data should be retrieved.

Returns

A reference to the data whose key is equal to __k, if such a data is present in the unordered_map.

Exceptions

std::out_of_range	If no such data is present.

begin() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_map.

begin() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin ()  [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the unordered_map.

begin() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
            size_type __n)  [inline]
```

Returns a read/write iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read/write local iterator.

begin() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

bucket_count()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count () const  [inline],
[noexcept]
```

Returns the number of buckets of the unordered_map.

cbegin() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_map.

cbegin() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

cend() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_map.

cend() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::cend (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

clear()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in an unordered_map. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
            const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void(), true)  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

contains() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

count()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Key to count.

Returns

Number of elements with specified key.

This function only makes sense for unordered_multimap; for unordered_map the result will either be 0 (not present) or 1 (present).

emplace()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename...  _Args>
std::pair< iterator, bool > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace (
            _Args &&...  __args)  [inline]
```

Attempts to build and insert a std::pair into the unordered_map.

Parameters

__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to build and insert a (key, value) pair into the unordered_map. An unordered_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered_map.
Insertion requires amortized constant time.

emplace_hint()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename...  _Args>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&...  __args)  [inline]
```

Attempts to build and insert a std::pair into the unordered_map.

Parameters

__pos	An iterator that serves as a hint as to where the pair should be inserted.
__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the element with key of the std::pair built from *__args* (may or may not be that std::pair).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↩associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

empty()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
bool std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::empty () const  [inline], [nodiscard],
[noexcept]
```
Returns true if the unordered_map is empty.

end() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end () const  [inline],
[noexcept]
```
Returns a read-only (constant) iterator that points one past the last element in the unordered_map.

end() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end ()  [inline], [noexcept]
```
Returns a read/write iterator that points one past the last element in the unordered_map.

end() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
            size_type __n)  [inline]
```
Returns a read/write iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read/write local iterator.

end() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
            size_type __n) const  [inline]
```
Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

equal_range() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, iterator > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::equal_↩
range (
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered_multimap.

equal_range() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< const_iterator, const_iterator > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc
>::equal_range (
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for unordered_multimap.

erase() [1/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<__heterogeneous_hash_key< unordered_map > _Kt>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            _Kt && __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_map. For an unordered_map the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_map. For an unordered_map the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [__first,__last) range of elements from an unordered_map.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from an unordered_map. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [4/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from an unordered_map.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [5/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            iterator __position)  [inline]
```

Erases an element from an unordered_map.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_map. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in an unordered_map.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in an unordered_map.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
allocator_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the unordered_map.

hash_function()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
hasher std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function () const  [inline]
```

Returns the hash functor object with which the unordered_map was constructed.

insert() [1/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, pair< iterator, bool > > std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
             _Pair && __x)  [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

_↩ *_x*	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered_map. An unordered_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered_map.
Insertion requires amortized constant time.

insert() [3/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, bool > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

_↩ *_x*	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered_map. An unordered_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered_map.
Insertion requires amortized constant time.

insert() [4/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::unordered_map<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            _Pair && __x)  [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↩associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

insert() [5/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            const value_type & __x)  [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↩associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

insert() [6/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            value_type && __x)  [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↩associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

insert() [7/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Attempts to insert a list of elements into the unordered_map.

Parameters

↩ _↩ ↩ _↩ *l*	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

insert() [8/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, bool > std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            value_type && __x)  [inline]
```

Attempts to insert a std::pair into the unordered_map.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

A pair, of which the first element is an iterator that points to the possibly inserted pair, and the second is a bool that is true if the pair was actually inserted.

This function attempts to insert a (key, value) pair into the unordered_map. An unordered_map relies on unique keys and thus a pair is only inserted if its first element (the key) is not already present in the unordered_map.
Insertion requires amortized constant time.

key_eq()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
key_equal std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq () const  [inline]
```

Returns the key comparison object with which the unordered_map was constructed.

load_factor()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor () const  [inline],
[noexcept]
```

Returns the average number of elements per bucket.

max_bucket_count()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count () const  [inline],
[noexcept]
```

Returns the maximum number of buckets of the unordered_map.

max_load_factor() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor () const  [inline],
[noexcept]
```

Returns a positive number that the unordered_map tries to keep the load factor less than or equal to.

max_load_factor() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (
            float __z)  [inline]
```

Change the unordered_map maximum load factor.

Parameters

_↩ _z	The new maximum load factor.

max_size()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::max_size () const  [inline],
[noexcept]
```

Returns the maximum size of the unordered_map.

operator=() [1/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
            const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & )  [default]
```

Copy assignment operator.

operator=() [2/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Unordered_map list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills an unordered_map with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the unordered_map and that the resulting unordered_map's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_map & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
            unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > && )  [default]
```

Move assignment operator.

operator[]() [1/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
template<__heterogeneous_hash_key< unordered_map > _Kt>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (
            _Kt && __k)  [inline]
```

Subscript ([]) access to unordered_map data.

Parameters

_↩ _k	The key for which data should be retrieved.

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([])operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.
Lookup requires constant time.

operator[]() [2/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (
            const key_type & __k)  [inline]
```

Subscript ([]) access to unordered_map data.

Parameters

_↩ _k	The key for which data should be retrieved.

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([])operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.
Lookup requires constant time.

operator[]() [3/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
mapped_type & std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::operator[] (
            key_type && __k)  [inline]
```

Subscript ([]) access to unordered_map data.

Parameters

_↩ _k	The key for which data should be retrieved.

Returns

A reference to the data of the (key,data) pair.

Allows for easy lookup with the subscript ([])operator. Returns data associated with the key specified in subscript. If the key does not exist, a pair with that key is created using default values, which is then returned.
Lookup requires constant time.

rehash()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::rehash (
            size_type __n)  [inline]
```

May rehash the unordered_map.

Parameters

_↩ _n	The new number of buckets.

Rehash will occur only if the new number of buckets respect the unordered_map maximum load factor.

reserve()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::reserve (
            size_type __n)  [inline]
```

Prepare the unordered_map for a specified number of elements.

Parameters

_↩ _n	Number of elements required.

Same as rehash(ceil(n / max_load_factor())).

size()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::size () const  [inline], [noexcept]
```

Returns the size of the unordered_map.

swap()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >::swap (
            unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another unordered_map.

Parameters

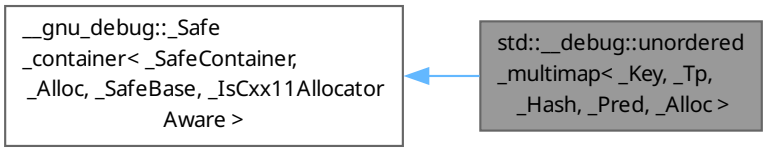
_↩ _x	An unordered_map of the same element and allocator types.

This exchanges the elements between two unordered_map in constant time. Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.
The documentation for this class was generated from the following file:

- unordered_map.h

6.1026 std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_map>`
Inheritance diagram for std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_multimap > **const_iterator**
- typedef __gnu_debug::_Safe_local_iterator< _Base_const_local_iterator, unordered_multimap > **const_local_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, unordered_multimap > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef __gnu_debug::_Safe_local_iterator< _Base_local_iterator, unordered_multimap > **local_iterator**
- typedef _Base::mapped_type **mapped_type**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_multimap** (_Base_ref __x)
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, const allocator_type &__a)

- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__↩ hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** (const allocator_type &__a)
- **unordered_multimap** (const unordered_multimap &)=default
- **unordered_multimap** (const unordered_multimap &__umap, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** (size_type __n, const allocator_type &__a)
- **unordered_multimap** (size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_multimap** (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** (unordered_multimap &&)=default
- **unordered_multimap** (unordered_multimap &&__umap, const allocator_type &__a) noexcept(noexcept(_↩ Base(std::move(__umap), __a)))
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- local_iterator **begin** (size_type __b)
- const_local_iterator **begin** (size_type __b) const
- size_type **bucket_size** (size_type __b) const
- const_iterator **cbegin** () const noexcept
- const_local_iterator **cbegin** (size_type __b) const
- const_iterator **cend** () const noexcept
- const_local_iterator **cend** (size_type __b) const
- void **clear** () noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __hint, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- local_iterator **end** (size_type __b)
- const_local_iterator **end** (size_type __b) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- _Base_iterator **erase** (_Base_const_iterator __it)
- template<__heterogeneous_hash_key< unordered_multimap > _Kt>
  size_type **erase** (_Kt &&__key)
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __first, const_iterator __last)

- iterator **erase** (const_iterator __it)
- iterator **erase** (iterator __it)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (_Pair &&__obj)
- iterator **insert** (const value_type &__obj)
- template<typename _Pair, typename = typename std::enable_if<std::is_constructible<value_type, _Pair&&>::value>::type>
  iterator **insert** (const_iterator __hint, _Pair &&__obj)
- iterator **insert** (const_iterator __hint, const value_type &__obj)
- iterator **insert** (const_iterator __hint, value_type &&__x)
- void **insert** (std::initializer_list< value_type > __l)
- iterator **insert** (value_type &&__x)
- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __f)
- unordered_multimap & **operator=** (const unordered_multimap &)=default
- unordered_multimap & **operator=** (initializer_list< value_type > __l)
- unordered_multimap & **operator=** (unordered_multimap &&)=default
- void **swap** (unordered_multimap &__x) noexcept(noexcept(declval< _Base & >().swap(__x)))

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**
- template<typename _ItT, typename _SeqT>
  class **::__gnu_debug::_Safe_local_iterator**

6.1026.1 Detailed Description

template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> > >
class std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

Class std::unordered_multimap with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/unordered_map

6.1027 std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_map>`

Public Types

- typedef _Hashtable::key_type key_type
- typedef _Hashtable::value_type value_type
- typedef _Hashtable::mapped_type mapped_type
- typedef _Hashtable::hasher hasher
- typedef _Hashtable::key_equal key_equal
- typedef _Hashtable::allocator_type allocator_type

- typedef _Hashtable::pointer pointer
- typedef _Hashtable::const_pointer const_pointer
- typedef _Hashtable::reference reference
- typedef _Hashtable::const_reference const_reference
- typedef _Hashtable::iterator iterator
- typedef _Hashtable::const_iterator const_iterator
- typedef _Hashtable::local_iterator local_iterator
- typedef _Hashtable::const_local_iterator const_local_iterator
- typedef _Hashtable::size_type size_type
- typedef _Hashtable::difference_type difference_type

Public Member Functions

- unordered_multimap ()=default
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multimap (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_multimap (const allocator_type &__a)
- unordered_multimap (const unordered_multimap &)=default
- **unordered_multimap** (const unordered_multimap &__ummap, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_multimap** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_multimap (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multimap** (size_type __n, const allocator_type &__a)
- **unordered_multimap** (size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_multimap (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_multimap (unordered_multimap &&)=default
- **unordered_multimap** (unordered_multimap &&__ummap, const allocator_type &__a) noexcept(noexcept(_↩ Hashtable(std::move(__ummap._M_h), __a)))
- iterator begin () noexcept

- local_iterator begin (size_type __n)
- size_type bucket_count () const noexcept
- size_type **bucket_size** (size_type __n) const
- void clear () noexcept
- template<typename... _Args>
  iterator emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- iterator end () noexcept
- local_iterator end (size_type __n)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- hasher hash_function () const
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_equal key_eq () const
- float load_factor () const noexcept
- size_type max_bucket_count () const noexcept
- float max_load_factor () const noexcept
- void max_load_factor (float __z)
- size_type max_size () const noexcept
- unordered_multimap & operator= (const unordered_multimap &)=default
- unordered_multimap & operator= (initializer_list< value_type > __l)
- unordered_multimap & operator= (unordered_multimap &&)=default
- void rehash (size_type __n)
- void reserve (size_type __n)
- size_type size () const noexcept
- void swap (unordered_multimap &__x) noexcept(noexcept(_M_h.swap(__x._M_h)))

- const_iterator begin () const noexcept
- const_iterator cbegin () const noexcept

- const_iterator end () const noexcept
- const_iterator cend () const noexcept

- iterator insert (const value_type &__x)
- iterator insert (value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > insert (_Pair &&__x)

- iterator insert (const_iterator __hint, const value_type &__x)
- iterator insert (const_iterator __hint, value_type &&__x)
- template<typename _Pair>
  __enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > insert (const_iterator __hint, _Pair &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __position)

- size_type erase (const key_type &__x)
- template<__heterogeneous_hash_key< unordered_multimap > _Kt>
  size_type erase (_Kt &&__x)

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const

- size_type count (const key_type &__x) const

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_h._M_find_tr(__x), void(), true)

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const

- size_type **bucket** (const key_type &__key) const
- template<__heterogeneous_hash_key< unordered_multimap > _Kt>
  size_type **bucket** (const _Kt &__key) const

- const_local_iterator begin (size_type __n) const
- const_local_iterator cbegin (size_type __n) const

- const_local_iterator end (size_type __n) const
- const_local_iterator cend (size_type __n) const

Friends

- template<typename _Key1, typename _Tp1, typename _Hash1, typename _Pred1, typename _Alloc1>
  bool **operator==** (const unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &, const unordered_multimap< _Key1, _Tp1, _Hash1, _Pred1, _Alloc1 > &)

6.1027.1 Detailed Description

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

A standard container composed of equivalent keys (possibly containing multiple of each key value) that associates values of another type with the keys.

Since

C++11

Template Parameters

_Key	Type of key objects.
_Tp	Type of mapped objects.
_Hash	Hashing function object type, defaults to hash<_Key>.
_Pred	Predicate function object type, defaults to equal_to<_Key>.
_Alloc	Allocator type, defaults to std::allocator<std::pair<const _Key, _Tp>>.

Meets the requirements of a container, and unordered associative container
The resulting value type of the container is std::pair<const _Key, _Tp>.
Base is _Hashtable, dispatched at compile time via template alias __ummap_hashtable.

6.1027.2 Member Typedef Documentation

allocator_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::allocator_type
```

Public typedefs.

const_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::const_iterator
```

Iterator-related typedefs.

const_local_iterator

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::const_local_iterator
```

Iterator-related typedefs.

const_pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

typedef _Hashtable::const_pointer std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::↩
const_pointer
Iterator-related typedefs.

const_reference

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::const_reference std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
>::const_reference
Iterator-related typedefs.

difference_type

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::difference_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc
>::difference_type
Iterator-related typedefs.

hasher

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hasher
Public typedefs.

iterator

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::iterator
Iterator-related typedefs.

key_equal

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_↩
equal
Public typedefs.

key_type

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::key_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_type
Public typedefs.

local_iterator

template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::↩
local_iterator
Iterator-related typedefs.

mapped_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::mapped_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::↩
mapped_type
```

Public typedefs.

pointer

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::pointer std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

reference

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::reference std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::↩
reference
```

Iterator-related typedefs.

size_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size_↩
type
```

Iterator-related typedefs.

value_type

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
typedef _Hashtable::value_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::↩
value_type
```

Public typedefs.

6.1027.3 Constructor & Destructor Documentation

unordered_multimap() [1/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap ()  [default]
```

Default constructor.

unordered_multimap() [2/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
            size_type __n,
            const hasher & __hf = hasher(),
```

```
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Default constructor creates no elements.

Parameters

__n	Mnimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

unordered_multimap() [3/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
            _InputIterator __first,
            _InputIterator __last,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_multimap from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_multimap consisting of copies of the elements from [__first,__last). This is linear in N (where N is distance(__first,__last)).

unordered_multimap() [4/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
            const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & )  [default]
```

Copy constructor.

unordered_multimap() [5/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
            unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > && )  [default]
```

Move constructor.

unordered_multimap() [6/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
            const allocator_type & __a)  [inline], [explicit]
```

Creates an unordered_multimap with no elements.

Parameters

_↩ _a	An allocator object.

unordered_multimap() [7/7]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::unordered_multimap (
            initializer_list< value_type > __l,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_multimap from an initializer_list.

Parameters

__l	An initializer_list.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_multimap consisting of copies of the elements in the list. This is linear in N (where N is *__l.size()*).

6.1027.4 Member Function Documentation

begin() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multimap.

begin() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin ()  [inline], [noexcept]
```

Returns a read/write iterator that points to the first element in the unordered_multimap.

begin() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
            size_type __n)  [inline]
```

Returns a read/write iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read/write local iterator.

begin() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::begin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

bucket_count()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::bucket_count () const  [inline],
[noexcept]
```

Returns the number of buckets of the unordered_multimap.

cbegin() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multimap.

cbegin() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cbegin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

cend() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cend () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multimap.

cend() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::cend (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

clear()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in an unordered_multimap. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Kt>
auto std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
            const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void(), true)  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

contains() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

count()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Key to count.

Returns

Number of elements with specified key.

emplace()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename...  _Args>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace (
            _Args &&...  __args)  [inline]
```

Attempts to build and insert a std::pair into the unordered_multimap.

Parameters

__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the inserted pair.

This function attempts to build and insert a (key, value) pair into the unordered_multimap.
Insertion requires amortized constant time.

emplace_hint()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename...  _Args>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&...  __args)  [inline]
```

Attempts to build and insert a std::pair into the unordered_multimap.

Parameters

__pos	An iterator that serves as a hint as to where the pair should be inserted.
__args	Arguments used to generate a new pair instance (see std::piecewise_contruct for passing arguments to each part of the pair constructor).

Returns

An iterator that points to the element with key of the std::pair built from *__args*.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

empty()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
bool std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::empty () const  [inline], [nodiscard], [noexcept]
```

Returns true if the unordered_multimap is empty.

end() [1/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multimap.

end() [2/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end ()  [inline], [noexcept]
```

Returns a read/write iterator that points one past the last element in the unordered_multimap.

end() [3/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
            size_type __n)  [inline]
```

Returns a read/write iterator pointing to one past the last bucket elements.

Parameters

__n	The bucket index.

Returns

A read/write local iterator.

end() [4/4]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
const_local_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::end (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

equal_range() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< iterator, iterator > std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::↩
equal_range (
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

equal_range() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
std::pair< const_iterator, const_iterator > std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _↩
Alloc >::equal_range (
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

erase() [1/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<__heterogeneous_hash_key< unordered_multimap > _Kt>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            _Kt && __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of elements to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of elements to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [__first,__last) range of elements from an unordered_multimap.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from an unordered_multimap. Note that this function only erases the elements, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [4/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from an unordered_multimap.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [5/5]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::erase (
            iterator __position)  [inline]
```

Erases an element from an unordered_multimap.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multimap. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in an unordered_multimap.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
const_iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in an unordered_multimap.

Parameters

_↩ _x	Key to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
allocator_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the unordered_multimap.

hash_function()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
hasher std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::hash_function () const  [inline]
```

Returns the hash functor object with which the unordered_multimap was constructed.

insert() [1/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _InputIterator>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            _Pair && __x)  [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

_↩ *_x*	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

insert() [3/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

_↩ *_x*	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

insert() [4/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
template<typename _Pair>
__enable_if_t< is_constructible< value_type, _Pair && >::value, iterator > std::unordered_multimap<
_Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            _Pair && __x)  [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.↩associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

insert() [5/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            const value_type & __x)  [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

insert() [6/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            value_type && __x)  [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

__hint	An iterator that serves as a hint as to where the pair should be inserted.
__x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the element with key of *__x* (may or may not be the pair passed in).

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints for more on *hinting*.
Insertion requires amortized constant time.

insert() [7/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Attempts to insert a list of elements into the unordered_multimap.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

insert() [8/8]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
iterator std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::insert (
            value_type && __x)  [inline]
```

Inserts a std::pair into the unordered_multimap.

Parameters

_↩ _x	Pair to be inserted (see std::make_pair for easy creation of pairs).

Returns

An iterator that points to the inserted pair.

Insertion requires amortized constant time.

key_eq()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
key_equal std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::key_eq () const  [inline]
```

Returns the key comparison object with which the unordered_multimap was constructed.

load_factor()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::load_factor () const  [inline],
[noexcept]
```

Returns the average number of elements per bucket.

max_bucket_count()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_bucket_count () const
[inline], [noexcept]
```

Returns the maximum number of buckets of the unordered_multimap.

max_load_factor() [1/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
float std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor () const  [inline],
[noexcept]
```

Returns a positive number that the unordered_multimap tries to keep the load factor less than or equal to.

max_load_factor() [2/2]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
```

```
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_load_factor (
            float __z)  [inline]
```

Change the unordered_multimap maximum load factor.

Parameters

_↩ _z	The new maximum load factor.

max_size()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::max_size () const  [inline],
[noexcept]
```

Returns the maximum size of the unordered_multimap.

operator=() [1/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap & std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
            const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & )  [default]
```

Copy assignment operator.

operator=() [2/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap & std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Unordered_multimap list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills an unordered_multimap with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the unordered_multimap and that the resulting unordered_multimap's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
unordered_multimap & std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::operator= (
            unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > && )  [default]
```

Move assignment operator.

rehash()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::rehash (
            size_type __n)  [inline]
```

May rehash the unordered_multimap.

Parameters

_↩ _n	The new number of buckets.

Rehash will occur only if the new number of buckets respect the unordered_multimap maximum load factor.

reserve()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::reserve (
            size_type __n)  [inline]
```

Prepare the unordered_multimap for a specified number of elements.

Parameters

_↩ _n	Number of elements required.

Same as rehash(ceil(n / max_load_factor())).

size()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
size_type std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::size () const  [inline],
[noexcept]
```

Returns the size of the unordered_multimap.

swap()

```
template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_↩
Key>, typename _Alloc = allocator<std::pair<const _Key, _Tp>>>
void std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >::swap (
            unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another unordered_multimap.

Parameters

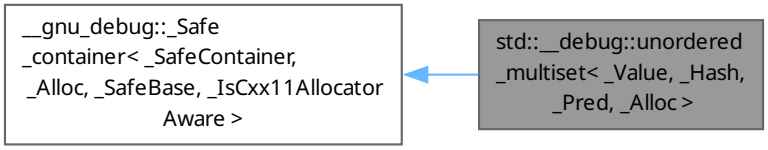
_↩ _x	An unordered_multimap of the same element and allocator types.

This exchanges the elements between two unordered_multimap in constant time. Note that the global std::swap() function is specialized such that std::swap(m1,m2) will feed to this function.
The documentation for this class was generated from the following file:

- unordered_map.h

6.1028 std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_set>`
Inheritance diagram for std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_multiset > **const_iterator**
- typedef __gnu_debug::_Safe_local_iterator< _Base_const_local_iterator, unordered_multiset > **const_local_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, unordered_multiset > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef __gnu_debug::_Safe_local_iterator< _Base_local_iterator, unordered_multiset > **local_iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_multiset** (_Base_ref __x)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (const allocator_type &__a)

- **unordered_multiset** (const unordered_multiset &)=default
- **unordered_multiset** (const unordered_multiset &__uset, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (size_type __n, const allocator_type &__a)
- **unordered_multiset** (size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_multiset** (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (unordered_multiset &&)=default
- **unordered_multiset** (unordered_multiset &&__uset, const allocator_type &__a) noexcept(noexcept(_↩ Base(std::move(__uset), __a)))
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- local_iterator **begin** (size_type __b)
- const_local_iterator **begin** (size_type __b) const
- size_type **bucket_size** (size_type __b) const
- const_iterator **cbegin** () const noexcept
- const_local_iterator **cbegin** (size_type __b) const
- const_iterator **cend** () const noexcept
- const_local_iterator **cend** (size_type __b) const
- void **clear** () noexcept
- template<typename... _Args>
  iterator **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __hint, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** (size_type __b)
- local_iterator **end** (size_type __b)
- const_local_iterator **end** (size_type __b) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- _Base_iterator **erase** (_Base_const_iterator __it)
- template<__heterogeneous_hash_key< unordered_multiset > _Kt>
  size_type **erase** (_Kt &&__key)
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __it)
- iterator **erase** (iterator __it)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- iterator **insert** (const value_type &__obj)
- iterator **insert** (const_iterator __hint, const value_type &__obj)
- iterator **insert** (const_iterator __hint, value_type &&__obj)

- void **insert** (std::initializer_list< value_type > __l)
- iterator **insert** (value_type &&__obj)
- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __f)
- unordered_multiset & **operator=** (const unordered_multiset &)=default
- unordered_multiset & **operator=** (initializer_list< value_type > __l)
- unordered_multiset & **operator=** (unordered_multiset &&)=default
- void **swap** (unordered_multiset &__x) noexcept(noexcept(declval< _Base & >().swap(__x)))

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**
- template<typename _ItT, typename _SeqT>
  class **::__gnu_debug::_Safe_local_iterator**

6.1028.1 Detailed Description

template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >

Class std::unordered_multiset with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/unordered_set

6.1029 std::unordered_multiset< _Value, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_set>`

Public Types

- typedef _Hashtable::key_type key_type
- typedef _Hashtable::value_type value_type
- typedef _Hashtable::hasher hasher
- typedef _Hashtable::key_equal key_equal
- typedef _Hashtable::allocator_type allocator_type

- typedef _Hashtable::pointer pointer
- typedef _Hashtable::const_pointer const_pointer
- typedef _Hashtable::reference reference
- typedef _Hashtable::const_reference const_reference
- typedef _Hashtable::iterator iterator
- typedef _Hashtable::const_iterator const_iterator
- typedef _Hashtable::local_iterator local_iterator
- typedef _Hashtable::const_local_iterator const_local_iterator
- typedef _Hashtable::size_type size_type
- typedef _Hashtable::difference_type difference_type

Public Member Functions

- unordered_multiset ()=default
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_multiset (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_multiset (const allocator_type &__a)
- unordered_multiset (const unordered_multiset &)=default
- **unordered_multiset** (const unordered_multiset &__umset, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_multiset** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_multiset (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_multiset** (size_type __n, const allocator_type &__a)
- **unordered_multiset** (size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_multiset (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_multiset (unordered_multiset &&)=default
- **unordered_multiset** (unordered_multiset &&__umset, const allocator_type &__a) noexcept(noexcept(_↩ Hashtable(std::move(__umset._M_h), __a)))
- size_type bucket_count () const noexcept
- size_type **bucket_size** (size_type __n) const
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- template<typename... _Args>
  iterator emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- template<__heterogeneous_hash_key< unordered_multiset > _Kt>
  size_type **erase** (_Kt &&__x)
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- hasher hash_function () const
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_equal key_eq () const
- float load_factor () const noexcept
- size_type max_bucket_count () const noexcept
- float max_load_factor () const noexcept

- void max_load_factor (float __z)
- size_type max_size () const noexcept
- unordered_multiset & operator= (const unordered_multiset &)=default
- unordered_multiset & operator= (initializer_list< value_type > __l)
- unordered_multiset & operator= (unordered_multiset &&)=default
- void rehash (size_type __n)
- void reserve (size_type __n)
- size_type size () const noexcept
- void swap (unordered_multiset &__x) noexcept(noexcept(_M_h.swap(__x._M_h)))

- iterator begin () noexcept
- const_iterator begin () const noexcept

- iterator end () noexcept
- const_iterator end () const noexcept

- iterator insert (const value_type &__x)
- iterator insert (value_type &&__x)

- iterator insert (const_iterator __hint, const value_type &__x)
- iterator insert (const_iterator __hint, value_type &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __position)

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const

- size_type count (const key_type &__x) const

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_h._M_find_tr(__x), void(), true)

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const

- size_type **bucket** (const key_type &__key) const

- template<__heterogeneous_hash_key< unordered_multiset > _Kt>
  size_type **bucket** (const _Kt &__key) const

- local_iterator begin (size_type __n)
- const_local_iterator begin (size_type __n) const
- const_local_iterator cbegin (size_type __n) const

- local_iterator end (size_type __n)
- const_local_iterator end (size_type __n) const
- const_local_iterator cend (size_type __n) const

Friends

- template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1>
  bool **operator==** (const unordered_multiset< _Value1, _Hash1, _Pred1, _Alloc1 > &, const unordered_multiset< _Value1, _Hash1, _Pred1, _Alloc1 > &)

6.1029.1 Detailed Description

template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>, typename _Alloc = allocator<_Value>>
class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >

A standard container composed of equivalent keys (possibly containing multiple of each key value) in which the elements' keys are the elements themselves.

Since

C++11

Template Parameters

_Value	Type of key objects.
_Hash	Hashing function object type, defaults to hash<_Value>.
_Pred	Predicate function object type, defaults to equal_to<_Value>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a container, and unordered associative container
Base is _Hashtable, dispatched at compile time via template alias __umset_hashtable.

6.1029.2 Member Typedef Documentation

allocator_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::↩
allocator_type
```

Public typedefs.

const_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::↩
const_iterator
```

Iterator-related typedefs.

const_local_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::const_local_iterator
```

Iterator-related typedefs.

const_pointer

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::↩
const_pointer
```

Iterator-related typedefs.

const_reference

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::↩
const_reference
```

Iterator-related typedefs.

difference_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::difference_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::↩
difference_type
```

Iterator-related typedefs.

hasher

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

key_equal

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

key_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

local_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::↩
local_iterator
```

Iterator-related typedefs.

pointer

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::pointer std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

reference

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::reference std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

size_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size_type
```

Iterator-related typedefs.

value_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::value_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::value_↩
type
```

Public typedefs.

6.1029.3 Constructor & Destructor Documentation

unordered_multiset() [1/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
```

```
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset ()  [default]
```

Default constructor.

unordered_multiset() [2/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
            size_type __n,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```

Default constructor creates no elements.

Parameters

__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

unordered_multiset() [3/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
            _InputIterator __first,
            _InputIterator __last,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_multiset from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_multiset consisting of copies of the elements from [__first,__last). This is linear in N (where N is distance(__first,__last)).

unordered_multiset() [4/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
            const unordered_multiset< _Value, _Hash, _Pred, _Alloc > & )  [default]
```

Copy constructor.

unordered_multiset() [5/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
            unordered_multiset< _Value, _Hash, _Pred, _Alloc > && )  [default]
```

Move constructor.

unordered_multiset() [6/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
            initializer_list< value_type > __l,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_multiset from an initializer_list.

Parameters

__l	An initializer_list.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_multiset consisting of copies of the elements in the list. This is linear in N (where N is *__l.size()*).

unordered_multiset() [7/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::unordered_multiset (
            const allocator_type & __a)  [inline], [explicit]
```

Creates an unordered_multiset with no elements.

Parameters

_↩ _a	An allocator object.

6.1029.4 Member Function Documentation

begin() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multiset.

begin() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin ()  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multiset.

begin() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (
            size_type __n)  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

begin() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::begin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

bucket_count()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::bucket_count () const  [inline],
[noexcept]
```

Returns the number of buckets of the unordered_multiset.

cbegin() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_multiset.

cbegin() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cbegin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

cend() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multiset.

cend() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::cend (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

clear()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in an unordered_multiset.
Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::contains (
            const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void(), true)  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

contains() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _*x*	Key of elements to be located.

Returns

True if there is any element with the specified key.

count()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements.

Parameters

_↩ _*x*	Element to located.

Returns

Number of elements with specified key.

emplace()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename...  _Args>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace (
            _Args &&...  __args)  [inline]
```

Builds and insert an element into the unordered_multiset.

Parameters

__args	Arguments used to generate an element.

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

emplace_hint()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename...  _Args>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::emplace_hint (
```

```
            const_iterator __pos,
            _Args &&... __args)  [inline]
```

Inserts an element into the unordered_multiset.

Parameters

__pos	An iterator that serves as a hint as to where the element should be inserted.
__args	Arguments used to generate the element to be inserted.

Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires amortized constant time.

empty()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::empty () const  [inline], [nodiscard],
[noexcept]
```

Returns true if the unordered_multiset is empty.

end() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multiset.

end() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end ()  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_multiset.

end() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (
            size_type __n)  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

end() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::end (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

equal_range() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, iterator > std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::equal↩
_range (
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

equal_range() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< const_iterator, const_iterator > std::unordered_multiset< _Value, _Hash, _Pred, _Alloc
>::equal_range (
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

erase() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_multiset.
Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [__first,__last) range of elements from an unordered_multiset.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from an unordered_multiset.
Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from an unordered_multiset.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multiset.
Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::erase (
            iterator __position)  [inline]
```

Erases an element from an unordered_multiset.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_multiset.
Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in an unordered_multiset.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in an unordered_multiset.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
allocator_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::get_allocator () const
[inline], [noexcept]
```

Returns the allocator object used by the unordered_multiset.

hash_function()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
hasher std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::hash_function () const  [inline]
```

Returns the hash functor object with which the unordered_multiset was constructed.

insert() [1/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that inserts a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Inserts an element into the unordered_multiset.

Parameters

_↩ _x	Element to be inserted.

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

insert() [3/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            const value_type & __x)  [inline]
```

Inserts an element into the unordered_multiset.

Parameters

__hint	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.↩ html#containers.associative.insert_hints
Insertion requires amortized constant.

insert() [4/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            value_type && __x)  [inline]
```

Inserts an element into the unordered_multiset.

Parameters

__hint	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the inserted element.

Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires amortized constant.

insert() [5/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Inserts a list of elements into the unordered_multiset.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

insert() [6/6]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::insert (
            value_type && __x)  [inline]
```

Inserts an element into the unordered_multiset.

Parameters

_↩ _x	Element to be inserted.

Returns

An iterator that points to the inserted element.

Insertion requires amortized constant time.

key_eq()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
key_equal std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::key_eq () const  [inline]
```

Returns the key comparison object with which the unordered_multiset was constructed.

load_factor()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::load_factor () const  [inline],
[noexcept]
```

Returns the average number of elements per bucket.

max_bucket_count()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_bucket_count () const
[inline], [noexcept]
```

Returns the maximum number of buckets of the unordered_multiset.

max_load_factor() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor () const  [inline],
[noexcept]
```

Returns a positive number that the unordered_multiset tries to keep the load factor less than or equal to.

max_load_factor() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_load_factor (
            float __z)  [inline]
```

Change the unordered_multiset maximum load factor.

Parameters

_↩ _z	The new maximum load factor.

max_size()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::max_size () const  [inline],
[noexcept]
```

Returns the maximum size of the unordered_multiset.

operator=() [1/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset & std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
            const unordered_multiset< _Value, _Hash, _Pred, _Alloc > & )  [default]
```

Copy assignment operator.

operator=() [2/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset & std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Unordered_multiset list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills an unordered_multiset with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the unordered_multiset and that the resulting unordered_multiset's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_multiset & std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::operator= (
            unordered_multiset< _Value, _Hash, _Pred, _Alloc > && )  [default]
```

Move assignment operator.

rehash()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::rehash (
            size_type __n)  [inline]
```

May rehash the unordered_multiset.

Parameters

_↩ _n	The new number of buckets.

Rehash will occur only if the new number of buckets respect the unordered_multiset maximum load factor.

reserve()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::reserve (
            size_type __n)  [inline]
```

Prepare the unordered_multiset for a specified number of elements.

Parameters

_↩ _n	Number of elements required.

Same as rehash(ceil(n / max_load_factor())).

size()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::size () const  [inline], [noexcept]
```

Returns the size of the unordered_multiset.

swap()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >::swap (
            unordered_multiset< _Value, _Hash, _Pred, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another unordered_multiset.

Parameters

_↩ _x	An unordered_multiset of the same element and allocator types.

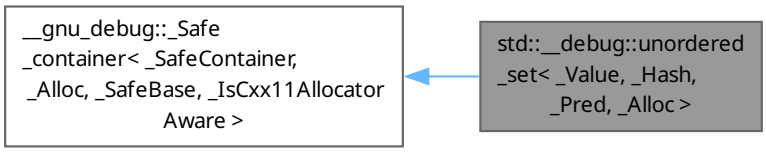
This exchanges the elements between two sets in constant time. Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.
The documentation for this class was generated from the following file:

- unordered_set.h

6.1030 std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference

```
#include <unordered_set>
```

Inheritance diagram for std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >:



Public Types

- typedef _Base::allocator_type **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, unordered_set > **const_iterator**
- typedef __gnu_debug::_Safe_local_iterator< _Base_const_local_iterator, unordered_set > **const_local_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef _Base::difference_type **difference_type**
- typedef _Base::hasher **hasher**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, unordered_set > **iterator**
- typedef _Base::key_equal **key_equal**
- typedef _Base::key_type **key_type**
- typedef __gnu_debug::_Safe_local_iterator< _Base_local_iterator, unordered_set > **local_iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef _Base::size_type **size_type**
- typedef _Base::value_type **value_type**

Public Member Functions

- **unordered_set** (_Base_ref __x)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** (const allocator_type &__a)
- **unordered_set** (const unordered_set &)=default
- **unordered_set** (const unordered_set &__uset, const allocator_type &__a)
- **unordered_set** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_set** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_set** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)

- **unordered_set** (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_↩ equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** (size_type __n, const allocator_type &__a)
- **unordered_set** (size_type __n, const hasher &__hf, const allocator_type &__a)
- **unordered_set** (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** (unordered_set &&)=default
- **unordered_set** (unordered_set &&__uset, const allocator_type &__a) noexcept(noexcept(_Base(std::move(_↩ _uset), __a)))
- const _Base & **_M_base** () const noexcept
- _Base & **_M_base** () noexcept
- const_iterator **begin** () const noexcept
- iterator **begin** () noexcept
- local_iterator **begin** (size_type __b)
- const_local_iterator **begin** (size_type __b) const
- size_type **bucket_size** (size_type __b) const
- const_iterator **cbegin** () const noexcept
- const_local_iterator **cbegin** (size_type __b) const
- const_iterator **cend** () const noexcept
- const_local_iterator **cend** (size_type __b) const
- void **clear** () noexcept
- template<typename... _Args>
  std::pair< iterator, bool > **emplace** (_Args &&... __args)
- template<typename... _Args>
  iterator **emplace_hint** (const_iterator __hint, _Args &&... __args)
- const_iterator **end** () const noexcept
- iterator **end** () noexcept
- local_iterator **end** (size_type __b)
- const_local_iterator **end** (size_type __b) const
- std::pair< iterator, iterator > **equal_range** (const key_type &__key)
- std::pair< const_iterator, const_iterator > **equal_range** (const key_type &__key) const
- _Base_iterator **erase** (_Base_const_iterator __it)
- template<__heterogeneous_hash_key< unordered_set > _Kt>
  size_type **erase** (_Kt &&__key)
- size_type **erase** (const key_type &__key)
- iterator **erase** (const_iterator __first, const_iterator __last)
- iterator **erase** (const_iterator __it)
- iterator **erase** (iterator __it)
- iterator **find** (const key_type &__key)
- const_iterator **find** (const key_type &__key) const
- template<typename _InputIterator>
  void **insert** (_InputIterator __first, _InputIterator __last)
- template<__heterogeneous_hash_key< unordered_set > _Kt>
  std::pair< iterator, bool > **insert** (_Kt &&__obj)
- std::pair< iterator, bool > **insert** (const value_type &__obj)
- template<__heterogeneous_hash_key< unordered_set > _Kt>
  iterator **insert** (const_iterator __hint, _Kt &&__obj)
- iterator **insert** (const_iterator __hint, const value_type &__obj)
- iterator **insert** (const_iterator __hint, value_type &&__obj)
- void **insert** (std::initializer_list< value_type > __l)
- std::pair< iterator, bool > **insert** (value_type &&__obj)

- float **max_load_factor** () const noexcept
- void **max_load_factor** (float __f)
- unordered_set & **operator=** (const unordered_set &)=default
- unordered_set & **operator=** (initializer_list< value_type > __l)
- unordered_set & **operator=** (unordered_set &&)=default
- void **swap** (unordered_set &__x) noexcept(noexcept(declval< _Base & >().swap(__x)))

Protected Member Functions

- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**
- template<typename _ItT, typename _SeqT>
  class **::__gnu_debug::_Safe_local_iterator**

6.1030.1 Detailed Description

template<typename _Value, typename _Hash = std::hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::allocator<_Value>>
class std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >

Class std::unordered_set with safety/checking/debug instrumentation.
The documentation for this class was generated from the following file:

- debug/unordered_set

6.1031 std::unordered_set< _Value, _Hash, _Pred, _Alloc > Class Template Reference

`#include <unordered_set>`

Public Types

- typedef _Hashtable::key_type key_type
- typedef _Hashtable::value_type value_type
- typedef _Hashtable::hasher hasher
- typedef _Hashtable::key_equal key_equal
- typedef _Hashtable::allocator_type allocator_type

- typedef _Hashtable::pointer pointer
- typedef _Hashtable::const_pointer const_pointer
- typedef _Hashtable::reference reference
- typedef _Hashtable::const_reference const_reference
- typedef _Hashtable::iterator iterator
- typedef _Hashtable::const_iterator const_iterator
- typedef _Hashtable::local_iterator local_iterator
- typedef _Hashtable::const_local_iterator const_local_iterator
- typedef _Hashtable::size_type size_type
- typedef _Hashtable::difference_type difference_type

Public Member Functions

- unordered_set ()=default
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, size_type __n, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, size_type __n, const hasher &__hf, const allocator_type &__a)
- template<typename _InputIterator>
  unordered_set (_InputIterator __first, _InputIterator __last, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_set (const allocator_type &__a)
- unordered_set (const unordered_set &)=default
- **unordered_set** (const unordered_set &__uset, const allocator_type &__a)
- **unordered_set** (initializer_list< value_type > __l, const allocator_type &__a)
- **unordered_set** (initializer_list< value_type > __l, size_type __n, const allocator_type &__a)
- **unordered_set** (initializer_list< value_type > __l, size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_set (initializer_list< value_type > __l, size_type __n=0, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- **unordered_set** (size_type __n, const allocator_type &__a)
- **unordered_set** (size_type __n, const hasher &__hf, const allocator_type &__a)
- unordered_set (size_type __n, const hasher &__hf=hasher(), const key_equal &__eql=key_equal(), const allocator_type &__a=allocator_type())
- unordered_set (unordered_set &&)=default
- **unordered_set** (unordered_set &&__uset, const allocator_type &__a) noexcept(noexcept(_Hashtable(std::move(↩
  __uset._M_h), __a)))
- size_type bucket_count () const noexcept
- size_type **bucket_size** (size_type __n) const
- const_iterator cbegin () const noexcept
- const_iterator cend () const noexcept
- void clear () noexcept
- template<typename... _Args>
  std::pair< iterator, bool > emplace (_Args &&... __args)
- template<typename... _Args>
  iterator emplace_hint (const_iterator __pos, _Args &&... __args)
- bool empty () const noexcept
- template<__heterogeneous_hash_key< unordered_set > _Kt>
  size_type **erase** (_Kt &&__x)
- size_type erase (const key_type &__x)
- iterator erase (const_iterator __first, const_iterator __last)
- allocator_type get_allocator () const noexcept
- hasher hash_function () const
- template<typename _InputIterator>
  void insert (_InputIterator __first, _InputIterator __last)
- void insert (initializer_list< value_type > __l)
- key_equal key_eq () const
- float load_factor () const noexcept
- size_type max_bucket_count () const noexcept
- float max_load_factor () const noexcept
- void max_load_factor (float __z)

- size_type max_size () const noexcept
- unordered_set & operator= (const unordered_set &)=default
- unordered_set & operator= (initializer_list< value_type > __l)
- unordered_set & operator= (unordered_set &&)=default
- void rehash (size_type __n)
- void reserve (size_type __n)
- size_type size () const noexcept
- void swap (unordered_set &__x) noexcept(noexcept(_M_h.swap(__x._M_h)))

- iterator begin () noexcept
- const_iterator begin () const noexcept

- iterator end () noexcept
- const_iterator end () const noexcept

- std::pair< iterator, bool > insert (const value_type &__x)
- std::pair< iterator, bool > insert (value_type &&__x)
- template<__heterogeneous_hash_key< unordered_set > _Kt>
  std::pair< iterator, bool > insert (_Kt &&__x)

- iterator insert (const_iterator __hint, const value_type &__x)
- iterator insert (const_iterator __hint, value_type &&__x)
- template<__heterogeneous_hash_key< unordered_set > _Kt>
  iterator insert (const_iterator, _Kt &&__x)

- iterator erase (const_iterator __position)
- iterator erase (iterator __position)

- iterator find (const key_type &__x)
- const_iterator find (const key_type &__x) const

- size_type count (const key_type &__x) const

- bool contains (const key_type &__x) const
- template<typename _Kt>
  auto contains (const _Kt &__x) const -> decltype(_M_h._M_find_tr(__x), void(), true)

- std::pair< iterator, iterator > equal_range (const key_type &__x)
- std::pair< const_iterator, const_iterator > equal_range (const key_type &__x) const

- size_type **bucket** (const key_type &__key) const
- template<__heterogeneous_hash_key< unordered_set > _Kt>
  size_type **bucket** (const _Kt &__key) const

- local_iterator begin (size_type __n)
- const_local_iterator begin (size_type __n) const
- const_local_iterator cbegin (size_type __n) const

- local_iterator end (size_type __n)
- const_local_iterator end (size_type __n) const
- const_local_iterator cend (size_type __n) const

Friends

- template<typename _Value1, typename _Hash1, typename _Pred1, typename _Alloc1>
  bool **operator==** (const unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 > &, const unordered_set< _Value1, _Hash1, _Pred1, _Alloc1 > &)

6.1031.1 Detailed Description

template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>, typename _Alloc = allocator<_Value>>
class std::unordered_set< _Value, _Hash, _Pred, _Alloc >

A standard container composed of unique keys (containing at most one of each key value) in which the elements' keys are the elements themselves.

Since

C++11

Template Parameters

_Value	Type of key objects.
_Hash	Hashing function object type, defaults to hash<_Value>.
_Pred	Predicate function object type, defaults to equal_to<_Value>.
_Alloc	Allocator type, defaults to allocator<_Key>.

Meets the requirements of a container, and unordered associative container
Base is _Hashtable, dispatched at compile time via template alias __uset_hashtable.

6.1031.2 Member Typedef Documentation

allocator_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::allocator_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::allocator↩
_type
```

Public typedefs.

const_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_↩
iterator
```

Iterator-related typedefs.

const_local_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::↩
const_local_iterator
```

Iterator-related typedefs.

const_pointer

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_pointer std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_↩
pointer
```

Iterator-related typedefs.

const_reference

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::const_reference std::unordered_set< _Value, _Hash, _Pred, _Alloc >::const_↩
reference
```

Iterator-related typedefs.

difference_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::difference_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::difference↩
_type
```

Iterator-related typedefs.

hasher

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::hasher std::unordered_set< _Value, _Hash, _Pred, _Alloc >::hasher
```

Public typedefs.

iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::iterator
```

Iterator-related typedefs.

key_equal

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_equal std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_equal
```

Public typedefs.

key_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::key_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_type
```

Public typedefs.

local_iterator

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::local_↩
iterator
```

Iterator-related typedefs.

pointer

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::pointer std::unordered_set< _Value, _Hash, _Pred, _Alloc >::pointer
```

Iterator-related typedefs.

reference

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::reference std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reference
```

Iterator-related typedefs.

size_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size_type
```

Iterator-related typedefs.

value_type

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
typedef _Hashtable::value_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::value_type
```

Public typedefs.

6.1031.3 Constructor & Destructor Documentation

unordered_set() [1/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set ()  [default]
```
Default constructor.

unordered_set() [2/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
            size_type __n,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline], [explicit]
```
Default constructor creates no elements.

Parameters

__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

unordered_set() [3/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _InputIterator>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
            _InputIterator __first,
            _InputIterator __last,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```
Builds an unordered_set from a range.

Parameters

__first	An input iterator.
__last	An input iterator.
__n	Minimal initial number of buckets.
__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_set consisting of copies of the elements from [__first,__last). This is linear in N (where N is distance(__first,__last)).

unordered_set() [4/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
            const unordered_set< _Value, _Hash, _Pred, _Alloc > & )  [default]
```

Copy constructor.

unordered_set() [5/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
            unordered_set< _Value, _Hash, _Pred, _Alloc > && )  [default]
```

Move constructor.

unordered_set() [6/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
            const allocator_type & __a)  [inline], [explicit]
```

Creates an unordered_set with no elements.

Parameters

_↩ _a	An allocator object.

unordered_set() [7/7]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::unordered_set< _Value, _Hash, _Pred, _Alloc >::unordered_set (
            initializer_list< value_type > __l,
            size_type __n = 0,
            const hasher & __hf = hasher(),
            const key_equal & __eql = key_equal(),
            const allocator_type & __a = allocator_type())  [inline]
```

Builds an unordered_set from an initializer_list.

Parameters

__l	An initializer_list.
__n	Minimal initial number of buckets.

__hf	A hash functor.
__eql	A key equality functor.
__a	An allocator object.

Create an unordered_set consisting of copies of the elements in the list. This is linear in N (where N is *__l.size()*).

6.1031.4 Member Function Documentation

begin() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_set.

begin() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin ()  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_set.

begin() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin (
            size_type __n)  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

begin() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::begin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

bucket_count()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::bucket_count () const  [inline],
[noexcept]
```

Returns the number of buckets of the unordered_set.

cbegin() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin () const  [inline],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the unordered_set.

cbegin() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cbegin (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to the first bucket element.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

cend() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cend () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_set.

cend() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::cend (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

clear()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::clear ()  [inline], [noexcept]
```

Erases all elements in an unordered_set. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

contains() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename _Kt>
auto std::unordered_set< _Value, _Hash, _Pred, _Alloc >::contains (
            const _Kt & __x) const -> decltype(_M_h._M_find_tr(__x), void(), true)  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

contains() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_set< _Value, _Hash, _Pred, _Alloc >::contains (
            const key_type & __x) const  [inline]
```

Finds whether an element with the given key exists.

Parameters

_↩ _x	Key of elements to be located.

Returns

True if there is any element with the specified key.

count()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::count (
            const key_type & __x) const  [inline]
```

Finds the number of elements.

Parameters

_↩ _x	Element to located.

Returns

Number of elements with specified key.

This function only makes sense for unordered_multisets; for unordered_set the result will either be 0 (not present) or 1 (present).

emplace()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename...  _Args>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::emplace (
            _Args &&...  __args)  [inline]
```

Attempts to build and insert an element into the unordered_set.

Parameters

__args	Arguments used to generate an element.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to build and insert an element into the unordered_set. An unordered_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered_set.
Insertion requires amortized constant time.

emplace_hint()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<typename... _Args>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::emplace_hint (
            const_iterator __pos,
            _Args &&... __args)  [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

__pos	An iterator that serves as a hint as to where the element should be inserted.
__args	Arguments used to generate the element to be inserted.

Returns

An iterator that points to the element with key equivalent to the one generated from *__args* (may or may not be the element itself).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument emplace() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires amortized constant time.

empty()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
bool std::unordered_set< _Value, _Hash, _Pred, _Alloc >::empty () const  [inline], [nodiscard],
[noexcept]
```

Returns true if the unordered_set is empty.

end() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end () const  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_set.

end() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end ()  [inline], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the unordered_set.

end() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end (
            size_type __n)  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

end() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_local_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::end (
            size_type __n) const  [inline]
```

Returns a read-only (constant) iterator pointing to one past the last bucket elements.

Parameters

_↩ _n	The bucket index.

Returns

A read-only local iterator.

equal_range() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, iterator > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::equal_range
(
            const key_type & __x)  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

equal_range() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
std::pair< const_iterator, const_iterator > std::unordered_set< _Value, _Hash, _Pred, _Alloc
>::equal_range (
            const key_type & __x) const  [inline]
```

Finds a subsequence matching given key.

Parameters

_↩ _x	Key to be located.

Returns

Pair of iterators that possibly points to the subsequence matching given key.

This function probably only makes sense for multisets.

erase() [1/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
            const key_type & __x)  [inline]
```

Erases elements according to the provided key.

Parameters

_↩ _x	Key of element to be erased.

Returns

The number of elements erased.

This function erases all the elements located by the given key from an unordered_set. For an unordered_set the result of this function can only be 0 (not present) or 1 (present). Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline]
```

Erases a [__first,__last) range of elements from an unordered_set.

Parameters

__first	Iterator pointing to the start of the range to be erased.
__last	Iterator pointing to the end of the range to be erased.

Returns

The iterator *__last*.

This function erases a sequence of elements from an unordered_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [3/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
            const_iterator __position)  [inline]
```

Erases an element from an unordered_set.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [4/4]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::erase (
            iterator __position)  [inline]
```

Erases an element from an unordered_set.

Parameters

__position	An iterator pointing to the element to be erased.

Returns

An iterator pointing to the element immediately following *__position* prior to the element being erased. If no such element exists, end() is returned.

This function erases an element, pointed to by the given iterator, from an unordered_set. Note that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

find() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (
            const key_type & __x)  [inline]
```

Tries to locate an element in an unordered_set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

find() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
const_iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::find (
            const key_type & __x) const  [inline]
```

Tries to locate an element in an unordered_set.

Parameters

_↩ _x	Element to be located.

Returns

Iterator pointing to sought-after element, or end() if not found.

This function takes a key and tries to locate the element with which the key matches. If successful the function returns an iterator pointing to the sought after element. If unsuccessful it returns the past-the-end (`end()`) iterator.

get_allocator()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
allocator_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::get_allocator () const  [inline],
[noexcept]
```

Returns the allocator object used by the unordered_set.

hash_function()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
hasher std::unordered_set< _Value, _Hash, _Pred, _Alloc >::hash_function () const  [inline]
```

Returns the hash functor object with which the unordered_set was constructed.

insert() [1/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
template<typename _InputIterator>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            _InputIterator __first,
            _InputIterator __last)  [inline]
```

A template function that attempts to insert a range of elements.

Parameters

__first	Iterator pointing to the start of the range to be inserted.
__last	Iterator pointing to the end of the range.

Complexity similar to that of the range constructor.

insert() [2/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<__heterogeneous_hash_key< unordered_set > _Kt>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            _Kt && __x)  [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

_↩ _x	Element to be inserted.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered_set. An unordered_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered_set.
Insertion requires amortized constant time.

insert() [3/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            const value_type & __x)  [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

_↩ _x	Element to be inserted.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered_set. An unordered_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered_set.
Insertion requires amortized constant time.

insert() [4/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            const value_type & __x)  [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

__hint	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the element with key of *__x* (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires amortized constant.

insert() [5/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            const_iterator __hint,
            value_type && __x)  [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

__hint	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the element with key of __x (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires amortized constant.

insert() [6/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
template<__heterogeneous_hash_key< unordered_set > _Kt>
iterator std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            const_iterator ,
            _Kt && __x)  [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

__hint	An iterator that serves as a hint as to where the element should be inserted.
__x	Element to be inserted.

Returns

An iterator that points to the element with key of __x (may or may not be the element passed in).

This function is not concerned about whether the insertion took place, and thus does not return a boolean like the single-argument insert() does. Note that the first parameter is only a hint and can potentially improve the performance of the insertion process. A bad hint would cause no gains in efficiency.
For more on *hinting*, see: https://gcc.gnu.org/onlinedocs/libstdc++/manual/associative.html#containers.associative.insert_hints
Insertion requires amortized constant.

insert() [7/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            initializer_list< value_type > __l)  [inline]
```

Attempts to insert a list of elements into the unordered_set.

Parameters

__l	A std::initializer_list<value_type> of elements to be inserted.

Complexity similar to that of the range constructor.

insert() [8/8]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
std::pair< iterator, bool > std::unordered_set< _Value, _Hash, _Pred, _Alloc >::insert (
            value_type && __x)  [inline]
```

Attempts to insert an element into the unordered_set.

Parameters

_↩ _x	Element to be inserted.

Returns

A pair, of which the first element is an iterator that points to the possibly inserted element, and the second is a bool that is true if the element was actually inserted.

This function attempts to insert an element into the unordered_set. An unordered_set relies on unique keys and thus an element is only inserted if it is not already present in the unordered_set.
Insertion requires amortized constant time.

key_eq()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
key_equal std::unordered_set< _Value, _Hash, _Pred, _Alloc >::key_eq () const  [inline]
```

Returns the key comparison object with which the unordered_set was constructed.

load_factor()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_set< _Value, _Hash, _Pred, _Alloc >::load_factor () const  [inline], [noexcept]
```

Returns the average number of elements per bucket.

max_bucket_count()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_bucket_count () const  [inline],
[noexcept]
```

Returns the maximum number of buckets of the unordered_set.

max_load_factor() [1/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
float std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_load_factor () const  [inline],
[noexcept]
```

Returns a positive number that the unordered_set tries to keep the load factor less than or equal to.

max_load_factor() [2/2]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
```

```
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_load_factor (
            float __z)  [inline]
```

Change the unordered_set maximum load factor.

Parameters

_↩ _z	The new maximum load factor.

max_size()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::max_size () const  [inline], [noexcept]
```

Returns the maximum size of the unordered_set.

operator=() [1/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set & std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
            const unordered_set< _Value, _Hash, _Pred, _Alloc > & )  [default]
```

Copy assignment operator.

operator=() [2/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set & std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline]
```

Unordered_set list assignment operator.

Parameters

↩ _↩ ↩ _↩ l	An initializer_list.

This function fills an unordered_set with copies of the elements in the initializer list __l.
Note that the assignment completely changes the unordered_set and that the resulting unordered_set's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
unordered_set & std::unordered_set< _Value, _Hash, _Pred, _Alloc >::operator= (
            unordered_set< _Value, _Hash, _Pred, _Alloc > && )  [default]
```

Move assignment operator.

rehash()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::rehash (
            size_type __n)  [inline]
```

May rehash the unordered_set.

Parameters

_↩ _n	The new number of buckets.

Rehash will occur only if the new number of buckets respect the unordered_set maximum load factor.

reserve()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::reserve (
            size_type __n)  [inline]
```

Prepare the unordered_set for a specified number of elements.

Parameters

_↩ _n	Number of elements required.

Same as rehash(ceil(n / max_load_factor())).

size()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
size_type std::unordered_set< _Value, _Hash, _Pred, _Alloc >::size () const  [inline], [noexcept]
```

Returns the size of the unordered_set.

swap()

```
template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = equal_to<_Value>,
typename _Alloc = allocator<_Value>>
void std::unordered_set< _Value, _Hash, _Pred, _Alloc >::swap (
            unordered_set< _Value, _Hash, _Pred, _Alloc > & __x)  [inline], [noexcept]
```

Swaps data with another unordered_set.

Parameters

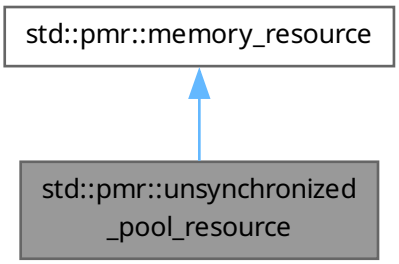
_↩ _x	An unordered_set of the same element and allocator types.

This exchanges the elements between two sets in constant time. Note that the global std::swap() function is specialized such that std::swap(s1,s2) will feed to this function.
The documentation for this class was generated from the following file:

- unordered_set.h

6.1032 std::pmr::unsynchronized_pool_resource Class Reference

`#include <memory_resource>`
Inheritance diagram for std::pmr::unsynchronized_pool_resource:



Public Member Functions

- **unsynchronized_pool_resource** (const pool_options &__opts)
- **unsynchronized_pool_resource** (const pool_options &__opts, memory_resource *__upstream)
- **unsynchronized_pool_resource** (const unsynchronized_pool_resource &)=delete
- **unsynchronized_pool_resource** (memory_resource *__upstream)
- void * **allocate** (size_t __bytes, size_t __alignment=_S_max_align)
- void **deallocate** (void *__p, size_t __bytes, size_t __alignment=_S_max_align)
- bool **is_equal** (const memory_resource &__other) const noexcept
- unsynchronized_pool_resource & **operator=** (const unsynchronized_pool_resource &)=delete
- pool_options **options** () const noexcept
- void **release** ()
- memory_resource * **upstream_resource** () const noexcept

Protected Member Functions

- void * do_allocate (size_t __bytes, size_t __alignment) override
- void do_deallocate (void *__p, size_t __bytes, size_t __alignment) override
- bool do_is_equal (const memory_resource &__other) const noexcept override

6.1032.1 Detailed Description

A non-thread-safe memory resource that manages pools of fixed-size blocks.

Since

C++17

6.1032.2 Member Function Documentation

do_allocate()

```
void * std::pmr::unsynchronized_pool_resource::do_allocate (
            size_t __bytes,
            size_t __alignment)  [override], [protected], [virtual]
```

Implements std::pmr::memory_resource.

do_deallocate()

```
void std::pmr::unsynchronized_pool_resource::do_deallocate (
            void * __p,
            size_t __bytes,
            size_t __alignment)  [override], [protected], [virtual]
```

Implements std::pmr::memory_resource.

do_is_equal()

```
bool std::pmr::unsynchronized_pool_resource::do_is_equal (
            const memory_resource & __other) const  [inline], [override], [protected], [virtual],
[noexcept]
```

Implements std::pmr::memory_resource.
The documentation for this class was generated from the following file:

- memory_resource

6.1033 std::uses_allocator< typename, typename > Struct Template Reference

6.1033.1 Detailed Description

template<typename, typename>
struct std::uses_allocator< typename, typename >

Declare uses_allocator so it can be specialized in <queue> etc.
The documentation for this struct was generated from the following file:

- memoryfwd.h

6.1034 std::uses_allocator< tuple< _Types... >, _Alloc > Struct Template Reference

```
#include <tuple>
```

6.1034.1 Detailed Description

template<typename... _Types, typename _Alloc>
struct std::uses_allocator< tuple< _Types... >, _Alloc >

Partial specialization for tuples.
The documentation for this struct was generated from the following file:

- tuple

6.1035 std::chrono::utc_clock Class Reference

```
#include <chrono>
```

Public Types

- using **duration**
- using **period**
- using **rep**
- using **time_point**

Static Public Member Functions

- template<typename _Duration>
  static utc_time< common_type_t< _Duration, seconds > > **from_sys** (const sys_time< _Duration > &__t)
- static time_point **now** ()
- template<typename _Duration>
  static sys_time< common_type_t< _Duration, seconds > > **to_sys** (const utc_time< _Duration > &__t)

Static Public Attributes

- static constexpr bool **is_steady**

6.1035.1 Detailed Description

A clock that measures Universal Coordinated Time (UTC).
The epoch is 1970-01-01 00:00:00.

Since

C++20

The documentation for this class was generated from the following file:

- chrono

6.1036 std::valarray< _Tp > Class Template Reference

`#include <valarray>`

Public Types

- typedef const _Tp * **const_iterator**
- typedef _Tp * **iterator**
- typedef _Tp **value_type**

Public Member Functions

- valarray () noexcept
- template<class _Dom>
  valarray (const _Expr< _Dom, _Tp > &__e)
- valarray (const _Tp &, size_t)
- template<typename _Tp>
  valarray (const _Tp *__restrict__ __p, size_t __n)
- valarray (const _Tp *__restrict__, size_t)
- valarray (const gslice_array< _Tp > &)
- valarray (const indirect_array< _Tp > &)
- valarray (const mask_array< _Tp > &)
- valarray (const slice_array< _Tp > &)
- valarray (const valarray &)

- valarray (initializer_list< _Tp >)
- valarray (size_t)
- valarray (valarray &&) noexcept
- _Expr< _ValFunClos< _ValArray, _Tp >, _Tp > apply (_Tp __func(_Tp)) const
- _Expr< _RefFunClos< _ValArray, _Tp >, _Tp > apply (_Tp __func(const _Tp &)) const
- const_iterator **begin** () const noexcept
- iterator begin () noexcept
- valarray< _Tp > cshift (int __n) const
- const_iterator **end** () const noexcept
- iterator end () noexcept
- _Tp max () const
- _Tp min () const
- _UnaryOp< __logical_not >::_Rt operator! () const
- template<class _Dom>
  valarray< _Tp > & **operator%=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator%= (const _Tp &)
- valarray< _Tp > & operator%= (const valarray< _Tp > &)
- template<class _Dom>
  valarray< _Tp > & **operator&=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator&= (const _Tp &)
- valarray< _Tp > & operator&= (const valarray< _Tp > &)
- template<class _Dom>
  valarray< _Tp > & **operator*=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator*= (const _Tp &)
- valarray< _Tp > & operator*= (const valarray< _Tp > &)
- _UnaryOp< __unary_plus >::_Rt operator+ () const
- template<class _Dom>
  valarray< _Tp > & **operator+=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator+= (const _Tp &)
- valarray< _Tp > & operator+= (const valarray< _Tp > &)
- _UnaryOp< __negate >::_Rt operator- () const
- template<class _Dom>
  valarray< _Tp > & **operator-=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator-= (const _Tp &)
- valarray< _Tp > & operator-= (const valarray< _Tp > &)
- template<class _Dom>
  valarray< _Tp > & **operator/=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator/= (const _Tp &)
- valarray< _Tp > & operator/= (const valarray< _Tp > &)
- template<class _Dom>
  valarray< _Tp > & **operator<<=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator<<= (const _Tp &)
- valarray< _Tp > & operator<<= (const valarray< _Tp > &)
- template<class _Dom>
  valarray< _Tp > & **operator=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator= (const _Tp &__t)
- valarray< _Tp > & operator= (const gslice_array< _Tp > &__ga)
- valarray< _Tp > & operator= (const indirect_array< _Tp > &__ia)
- valarray< _Tp > & operator= (const mask_array< _Tp > &__ma)
- valarray< _Tp > & operator= (const slice_array< _Tp > &__sa)
- valarray< _Tp > & operator= (const valarray< _Tp > &__v)

- valarray & operator= (initializer_list< _Tp > __l)
- valarray< _Tp > & operator= (valarray< _Tp > &&__v) noexcept
- template<class _Dom>
  valarray< _Tp > & **operator>>=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator>>= (const _Tp &)
- valarray< _Tp > & operator>>= (const valarray< _Tp > &)
- gslice_array< _Tp > operator[ ] (const gslice &__s)
- _Expr< _GClos< _ValArray, _Tp >, _Tp > operator[ ] (const gslice &__s) const
- mask_array< _Tp > operator[ ] (const valarray< bool > &__m)
- valarray< _Tp > operator[ ] (const valarray< bool > &__m) const
- indirect_array< _Tp > operator[ ] (const valarray< size_t > &__i)
- _Expr< _IClos< _ValArray, _Tp >, _Tp > operator[ ] (const valarray< size_t > &__i) const
- _Tp & operator[ ] (size_t __i) noexcept
- const _Tp & **operator[ ]** (size_t) const noexcept
- slice_array< _Tp > operator[ ] (slice __s)
- _Expr< _SClos< _ValArray, _Tp >, _Tp > operator[ ] (slice __s) const
- template<class _Dom>
  valarray< _Tp > & **operator^=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator^= (const _Tp &)
- valarray< _Tp > & operator^= (const valarray< _Tp > &)
- template<class _Dom>
  valarray< _Tp > & **operator|=** (const _Expr< _Dom, _Tp > &)
- valarray< _Tp > & operator|= (const _Tp &)
- valarray< _Tp > & operator|= (const valarray< _Tp > &)
- _UnaryOp< __bitwise_not >::_Rt operator~ () const
- void resize (size_t __size, _Tp __c=_Tp())
- valarray< _Tp > shift (int __n) const
- size_t size () const
- _Tp sum () const
- void swap (valarray< _Tp > &__v) noexcept

Friends

- struct **_Array< _Tp >**

6.1036.1 Detailed Description

template<class _Tp>
class std::valarray< _Tp >

Smart array designed to support numeric processing.
A valarray is an array that provides constraints intended to allow for effective optimization of numeric array processing by reducing the aliasing that can result from pointer representations. It represents a one-dimensional array from which different multidimensional subsets can be accessed and modified.

Template Parameters

_Tp	Type of object in the array.

6.1036.2 Constructor & Destructor Documentation

valarray()

```
template<class _Tp>
std::valarray< _Tp >::valarray (
            const _Tp * __restrict__,
            size_t )
```

Construct an array initialized to the first *n* elements of *t*.
References valarray().

6.1036.3 Member Function Documentation

begin()

```
template<class _Tp>
iterator std::valarray< _Tp >::begin ()  [inline], [nodiscard], [noexcept]
```

Return an iterator pointing to the first element of the valarray.

end()

```
template<class _Tp>
iterator std::valarray< _Tp >::end ()  [inline], [nodiscard], [noexcept]
```

Return an iterator pointing to one past the last element of the valarray.

operator"!()

```
template<class _Tp>
_UnaryOp< __logical_not >::_Rt std::valarray< _Tp >::operator!  () const
```

Return a new valarray by applying unary ! to each element.

operator%=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator%= (
            const _Tp & )
```

Set each element e of array to e % *t*.
References valarray().

operator%=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator%= (
            const valarray< _Tp > & )
```

Modulo elements of array by corresponding elements of *v*.
References valarray().

operator&=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator&= (
            const _Tp & )
```

Set each element e of array to e & *t*.
References valarray().

operator&=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator&= (
            const valarray< _Tp > & )
```

Logical and corresponding elements of *v* with elements of array.
References valarray().

operator*=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator*= (
            const _Tp & )
```

Multiply each element of array by *t*.
References valarray().

operator*=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator*= (
            const valarray< _Tp > & )
```

Multiply elements of array by corresponding elements of *v*.
References valarray().

operator+()

```
template<class _Tp>
_UnaryOp< __unary_plus >::_Rt std::valarray< _Tp >::operator+ () const
```

Return a new valarray by applying unary + to each element.

operator+=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator+= (
            const _Tp & )
```

Add *t* to each element of array.
References valarray().

operator+=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator+= (
            const valarray< _Tp > & )
```

Add corresponding elements of *v* to elements of array.
References valarray().

operator-()

```
template<class _Tp>
_UnaryOp< __negate >::_Rt std::valarray< _Tp >::operator- () const
```

Return a new valarray by applying unary - to each element.

operator-=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator-= (
            const _Tp & )
```

Subtract *t* to each element of array.
References valarray().

operator-=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator-= (
            const valarray< _Tp > & )
```

Subtract corresponding elements of *v* from elements of array.
References valarray().

operator/=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator/= (
            const _Tp & )
```

Divide each element of array by *t*.
References valarray().

operator/=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator/= (
            const valarray< _Tp > & )
```

Divide elements of array by corresponding elements of *v*.
References valarray().

operator<<=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator<<= (
            const _Tp & )
```

Left shift each element e of array by *t* bits.
References valarray().

operator<<=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator<<= (
            const valarray< _Tp > & )
```

Left shift elements of array by corresponding elements of *v*.

operator>>=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator>>= (
            const _Tp & )
```

Right shift each element e of array by *t* bits.
References valarray().

operator>>=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator>>= (
            const valarray< _Tp > & )
```

Right shift elements of array by corresponding elements of *v*.
References valarray().

operator^=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator^= (
            const _Tp & )
```

Set each element e of array to e ^ *t*.
References valarray().

operator^=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator^= (
            const valarray< _Tp > & )
```

Logical xor corresponding elements of *v* with elements of array.
References valarray().

operator"|=() [1/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator|= (
            const _Tp & )
```

Set each element e of array to e | *t*.

operator"|=() [2/2]

```
template<class _Tp>
valarray< _Tp > & std::valarray< _Tp >::operator|= (
            const valarray< _Tp > & )
```

Logical or corresponding elements of *v* with elements of array.
References valarray().

operator~()

```
template<class _Tp>
_UnaryOp< __bitwise_not >::_Rt std::valarray< _Tp >::operator~ () const
```

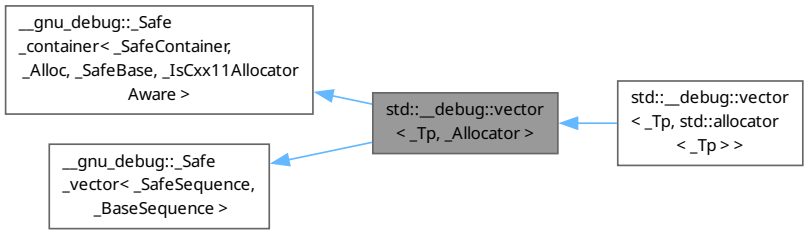
Return a new valarray by applying unary ~ to each element.
The documentation for this class was generated from the following file:

- valarray

6.1037 std::__debug::vector< _Tp, _Allocator > Class Template Reference

```
#include <vector>
```

Inheritance diagram for std::__debug::vector< _Tp, _Allocator >:



Public Types

- typedef _Allocator **allocator_type**
- typedef __gnu_debug::_Safe_iterator< _Base_const_iterator, vector > **const_iterator**
- typedef _Base::const_pointer **const_pointer**
- typedef _Base::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef _Base::difference_type **difference_type**
- typedef __gnu_debug::_Safe_iterator< _Base_iterator, vector > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Base::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef _Base::size_type **size_type**
- typedef _Tp **value_type**

Public Member Functions

- constexpr vector (_Base_ref __x)
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr **vector** (_InputIterator __first, _InputIterator __last, const _Allocator &__a=_Allocator())
- constexpr **vector** (const _Allocator &__a) noexcept
- **vector** (const vector &)=default
- constexpr **vector** (const vector &__x, const __type_identity_t< allocator_type > &__a)
- constexpr **vector** (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- constexpr **vector** (size_type __n, const __type_identity_t< _Tp > &__value, const _Allocator &__a=_Allocator())
- constexpr **vector** (size_type __n, const _Allocator &__a=_Allocator())
- **vector** (vector &&)=default
- constexpr **vector** (vector &&__x, const __type_identity_t< allocator_type > &__a) noexcept(std::is_nothrow_constructible< _Base, _Base, const allocator_type & >::value)
- constexpr const _Base & **_M_base** () const noexcept
- constexpr _Base & **_M_base** () noexcept
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr void **assign** (_InputIterator __first, _InputIterator __last)
- constexpr void **assign** (initializer_list< value_type > __l)
- constexpr void **assign** (size_type __n, const _Tp &__u)
- constexpr const_reference **back** () const noexcept
- constexpr reference **back** () noexcept

- constexpr const_iterator **begin** () const noexcept
- constexpr iterator **begin** () noexcept
- constexpr size_type **capacity** () const noexcept
- constexpr const_iterator **cbegin** () const noexcept
- constexpr const_iterator **cend** () const noexcept
- constexpr void **clear** () noexcept
- constexpr const_reverse_iterator **crbegin** () const noexcept
- constexpr const_reverse_iterator **crend** () const noexcept
- template<typename... _Args>
  constexpr iterator **emplace** (const_iterator __position, _Args &&... __args)
- template<typename... _Args>
  constexpr reference **emplace_back** (_Args &&... __args)
- constexpr const_iterator **end** () const noexcept
- constexpr iterator **end** () noexcept
- constexpr iterator **erase** (const_iterator __first, const_iterator __last)
- constexpr iterator **erase** (const_iterator __position)
- constexpr const_reference **front** () const noexcept
- constexpr reference **front** () noexcept
- template<class _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr iterator **insert** (const_iterator __position, _InputIterator __first, _InputIterator __last)
- template<typename _Up = _Tp>
  constexpr __gnu_cxx::__enable_if<!std::__are_same< _Up, bool >::__value, iterator >::__type **insert** (const↩
  _iterator __position, _Tp &&__x)
- constexpr iterator **insert** (const_iterator __position, const _Tp &__x)
- constexpr iterator **insert** (const_iterator __position, initializer_list< value_type > __l)
- constexpr iterator **insert** (const_iterator __position, size_type __n, const _Tp &__x)
- vector & **operator=** (const vector &)=default
- constexpr vector & **operator=** (initializer_list< value_type > __l)
- vector & **operator=** (vector &&)=default
- constexpr const_reference **operator[ ]** (size_type __n) const noexcept
- constexpr reference **operator[ ]** (size_type __n) noexcept
- constexpr void **pop_back** () noexcept
- template<typename _Up = _Tp>
  constexpr __gnu_cxx::__enable_if<!std::__are_same< _Up, bool >::__value, void >::__type **push_back** (_Tp
  &&__x)
- constexpr void **push_back** (const _Tp &__x)
- constexpr const_reverse_iterator **rbegin** () const noexcept
- constexpr reverse_iterator **rbegin** () noexcept
- constexpr const_reverse_iterator **rend** () const noexcept
- constexpr reverse_iterator **rend** () noexcept
- constexpr void **reserve** (size_type __n)
- constexpr void **resize** (size_type __sz)
- constexpr void **resize** (size_type __sz, const _Tp &__c)
- constexpr void **shrink_to_fit** ()
- constexpr void **swap** (vector &__x) noexcept(/*conditional */)

Protected Member Functions

- bool **_M_requires_reallocation** (size_type __elements) const noexcept
- constexpr void **_M_swap** (const _Safe_container &__x) const noexcept
- constexpr void **_M_update_guaranteed_capacity** () noexcept

Protected Attributes

- size_type **_M_guaranteed_capacity**

Friends

- template<typename _ItT, typename _SeqT, typename _CatT>
  class **::__gnu_debug::_Safe_iterator**

6.1037.1 Detailed Description

template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
class std::__debug::vector< _Tp, _Allocator >

Class std::vector with safety/checking/debug instrumentation.

6.1037.2 Constructor & Destructor Documentation

vector()

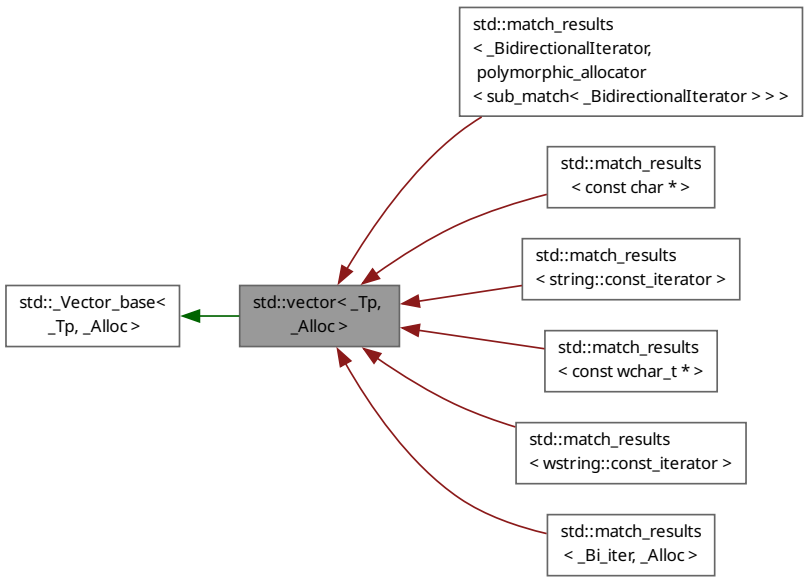
```
template<typename _Tp, typename _Allocator = std::allocator<_Tp>>
std::__debug::vector< _Tp, _Allocator >::vector (
            _Base_ref __x)  [inline], [constexpr]
```

Construction from a normal-mode vector.
The documentation for this class was generated from the following file:

- debug/vector

6.1038 std::vector< _Tp, _Alloc > Class Template Reference

`#include <vector>`
Inheritance diagram for std::vector< _Tp, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef __gnu_cxx::__normal_iterator< const_pointer, vector > **const_iterator**
- typedef _Alloc_traits::const_pointer **const_pointer**
- typedef _Alloc_traits::const_reference **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef __gnu_cxx::__normal_iterator< pointer, vector > **iterator**
- typedef _Base::pointer **pointer**
- typedef _Alloc_traits::reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef _Tp **value_type**

Public Member Functions

- vector ()=default
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr vector (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- constexpr vector (const allocator_type &__a) noexcept
- constexpr vector (const vector &__x)
- constexpr vector (const vector &__x, const __type_identity_t< allocator_type > &__a)
- constexpr vector (initializer_list< value_type > __l, const allocator_type &__a=allocator_type())
- constexpr vector (size_type __n, const allocator_type &__a=allocator_type())
- constexpr vector (size_type __n, const value_type &__value, const allocator_type &__a=allocator_type())
- vector (vector &&) noexcept=default
- constexpr vector (vector &&__rv, const __type_identity_t< allocator_type > &__m) noexcept(noexcept(vector(std::declval< vector && >(), std::declval< const allocator_type & >(), std::declval< typename _Alloc_traits::is_always_equal >())))
- constexpr ~vector () noexcept
- template<typename... _Args>
  constexpr auto **_M_emplace_aux** (const_iterator __position, _Args &&... __args) -> iterator
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr void assign (_InputIterator __first, _InputIterator __last)
- constexpr void assign (initializer_list< value_type > __l)
- constexpr void assign (size_type __n, const value_type &__val)
- constexpr reference at (size_type __n)
- constexpr const_reference at (size_type __n) const
- constexpr const_reference back () const noexcept
- constexpr reference back () noexcept
- constexpr const_iterator begin () const noexcept
- constexpr iterator begin () noexcept
- constexpr size_type capacity () const noexcept
- constexpr const_iterator cbegin () const noexcept
- constexpr const_iterator cend () const noexcept
- constexpr void clear () noexcept
- constexpr const_reverse_iterator crbegin () const noexcept
- constexpr const_reverse_iterator crend () const noexcept
- constexpr const _Tp * **data** () const noexcept
- constexpr _Tp * data () noexcept

- template<typename... _Args>
  constexpr iterator emplace (const_iterator __position, _Args &&... __args)
- template<typename... _Args>
  constexpr reference **emplace_back** (_Args &&... __args)
- constexpr bool empty () const noexcept
- constexpr const_iterator end () const noexcept
- constexpr iterator end () noexcept
- constexpr iterator erase (const_iterator __first, const_iterator __last)
- constexpr iterator erase (const_iterator __position)
- constexpr const_reference front () const noexcept
- constexpr reference front () noexcept
- constexpr allocator_type get_allocator () const noexcept
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr iterator insert (const_iterator __position, _InputIterator __first, _InputIterator __last)
- constexpr iterator insert (const_iterator __position, const value_type &__x)
- constexpr iterator insert (const_iterator __position, initializer_list< value_type > __l)
- constexpr iterator insert (const_iterator __position, size_type __n, const value_type &__x)
- constexpr iterator insert (const_iterator __position, value_type &&__x)
- constexpr size_type max_size () const noexcept
- constexpr vector & operator= (const vector &__x)
- constexpr vector & operator= (initializer_list< value_type > __l)
- constexpr vector & operator= (vector &&__x) noexcept(_Alloc_traits::_S_nothrow_move())
- constexpr const_reference operator[ ] (size_type __n) const noexcept
- constexpr reference operator[ ] (size_type __n) noexcept
- constexpr void pop_back () noexcept
- constexpr void push_back (const value_type &__x)
- constexpr void **push_back** (value_type &&__x)
- constexpr const_reverse_iterator rbegin () const noexcept
- constexpr reverse_iterator rbegin () noexcept
- constexpr const_reverse_iterator rend () const noexcept
- constexpr reverse_iterator rend () noexcept
- constexpr void reserve (size_type __n)
- constexpr void resize (size_type __new_size)
- constexpr void resize (size_type __new_size, const value_type &__x)
- constexpr void shrink_to_fit ()
- constexpr size_type size () const noexcept
- constexpr void swap (vector &__x) noexcept

Protected Member Functions

- constexpr pointer **_M_allocate** (size_t __n)
- template<typename _ForwardIterator>
  constexpr _Alloc_result _M_allocate_and_copy (size_type __n, _ForwardIterator __first, _ForwardIterator __last)
- constexpr _Alloc_result **_M_allocate_at_least** (size_t __n)
- template<typename _ForwardIterator>
  constexpr void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  constexpr void **_M_assign_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _InputIterator>
  constexpr void **_M_assign_dispatch** (_InputIterator __first, _InputIterator __last, __false_type)
- template<typename _Integer>
  constexpr void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)

- constexpr size_type **_M_check_len** (size_type __n, const char *__s) const
- constexpr void **_M_create_storage** (size_t __n)
- constexpr void **_M_deallocate** (pointer __p, size_t __n)
- constexpr void **_M_default_append** (size_type __n)
- constexpr void **_M_default_initialize** (size_type __n)
- template<typename... _Args>
  constexpr iterator **_M_emplace_aux** (const_iterator __position, _Args &&... __args)
- constexpr iterator **_M_emplace_aux** (const_iterator __position, value_type &&__v)
- constexpr iterator **_M_erase** (iterator __first, iterator __last)
- constexpr iterator **_M_erase** (iterator __position)
- constexpr void **_M_erase_at_end** (pointer __pos) noexcept
- constexpr void **_M_fill_append** (size_type __n, const value_type &__x)
- constexpr void **_M_fill_assign** (size_type __n, const value_type &__val)
- constexpr void **_M_fill_initialize** (size_type __n, const value_type &__value)
- constexpr void **_M_fill_insert** (iterator __pos, size_type __n, const value_type &__x)
- constexpr const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- constexpr _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- template<typename _Arg>
  constexpr void **_M_insert_aux** (iterator __position, _Arg &&__arg)
- template<typename _InputIterator>
  constexpr void **_M_insert_dispatch** (iterator __pos, _InputIterator __first, _InputIterator __last, __false_type)
- template<typename _Integer>
  constexpr void **_M_insert_dispatch** (iterator __pos, _Integer __n, _Integer __val, __true_type)
- constexpr iterator **_M_insert_rval** (const_iterator __position, value_type &&__v)
- constexpr void _M_range_check (size_type __n) const
- template<typename _ForwardIterator>
  constexpr void **_M_range_initialize** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  constexpr void **_M_range_initialize** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _Iterator, typename _Sentinel>
  constexpr void **_M_range_initialize_n** (_Iterator __first, _Sentinel __last, size_type __n)
- template<typename _ForwardIterator>
  constexpr void **_M_range_insert** (iterator __pos, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  constexpr void **_M_range_insert** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename... _Args>
  constexpr void **_M_realloc_append** (_Args &&... __args)
- template<typename... _Args>
  constexpr void **_M_realloc_insert** (iterator __position, _Args &&... __args)
- constexpr void **_M_replace_storage** (pointer __start, pointer __end, size_type __cap)
- template<typename _ForwardIterator>
  constexpr void **_M_replace_with** (size_type __n, _ForwardIterator __first, _ForwardIterator __last)
- constexpr bool **_M_shrink_to_fit** ()

Static Protected Member Functions

- static constexpr size_type **_S_check_init_len** (size_type __n, const allocator_type &__a)
- static constexpr size_t **_S_max_size** (const _Tp_alloc_type &__a) noexcept

Protected Attributes

- _Vector_impl **_M_impl**

6.1038.1 Detailed Description

template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
class std::vector< _Tp, _Alloc >

A standard container which offers fixed time access to individual elements in any order.

Since

C++98

Template Parameters

_Tp	Type of element.
_Alloc	Allocator type, defaults to allocator<_Tp>.

Meets the requirements of a container, a reversible container, and a sequence, including the optional sequence requirements with the exception of push_front and pop_front.
In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ([]) access is also provided as with C-style arrays.

6.1038.2 Constructor & Destructor Documentation

vector() [1/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector ()  [default]
```

Creates a vector with no elements.
Referenced by insert(), and operator=().

vector() [2/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            const allocator_type & __a)  [inline], [explicit], [constexpr], [noexcept]
```

Creates a vector with no elements.

Parameters

_↩ _a	An allocator object.

vector() [3/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            size_type __n,
            const allocator_type & __a = allocator_type())  [inline], [explicit], [constexpr]
```

Creates a vector with default constructed elements.

Parameters

_↩ _n	The number of elements to initially create.
_↩ _a	An allocator.

This constructor fills the vector with __n default constructed elements.

vector() [4/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            size_type __n,
            const value_type & __value,
            const allocator_type & __a = allocator_type())  [inline], [constexpr]
```

Creates a vector with copies of an exemplar element.

Parameters

__n	The number of elements to initially create.
__value	An element to copy.
__a	An allocator.

This constructor fills the vector with __n copies of __value.

vector() [5/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            const vector< _Tp, _Alloc > & __x)  [inline], [constexpr]
```

Vector copy constructor.

Parameters

_↩ _x	A vector of identical element and allocator types.

All the elements of __x are copied, but any unused capacity in __x will not be copied (i.e. capacity() == size() in the new vector).
The newly-created vector uses a copy of the allocator object used by __x (unless the allocator traits dictate a different object).

vector() [6/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            vector< _Tp, _Alloc > && )  [default], [noexcept]
```

Vector move constructor.
The newly-created vector contains the exact contents of the moved instance. The contents of the moved instance are a valid, but unspecified vector.

vector() [7/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            const vector< _Tp, _Alloc > & __x,
            const __type_identity_t< allocator_type > & __a)  [inline], [constexpr]
```

Copy constructor with alternative allocator.

vector() [8/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            vector< _Tp, _Alloc > && __rv,
            const __type_identity_t< allocator_type > & __m)  [inline], [constexpr], [noexcept]
```

Move constructor with alternative allocator.

vector() [9/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::vector (
            initializer_list< value_type > __l,
            const allocator_type & __a = allocator_type())  [inline], [constexpr]
```

Builds a vector from an initializer list.

Parameters

_↩ _*l*	An initializer_list.
_↩ _*a*	An allocator.

Create a vector consisting of copies of the elements in the initializer_list __*l*.
This will call the element type's copy constructor N times (where N is __*l.size()*) and do no memory reallocation.

vector() [10/10]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
std::vector< _Tp, _Alloc >::vector (
            _InputIterator __first,
            _InputIterator __last,
            const allocator_type & __a = allocator_type())  [inline], [constexpr]
```

Builds a vector from a range.

Parameters

__*first*	An input iterator.
__*last*	An input iterator.
__*a*	An allocator.

Create a vector consisting of copies of the elements from [first,last).
If the iterators are forward, bidirectional, or random-access, then this will call the elements' copy constructor N times (where N is distance(first,last)) and do no memory reallocation. But if only input iterators are used, then this will do at most 2N calls to the copy constructor, and logN memory reallocations.

~vector()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
std::vector< _Tp, _Alloc >::~vector ()  [inline], [constexpr], [noexcept]
```

The dtor only erases the elements, and note that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

6.1038.3 Member Function Documentation

_M_allocate_and_copy()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _ForwardIterator>
_Alloc_result std::vector< _Tp, _Alloc >::_M_allocate_and_copy (
            size_type __n,
            _ForwardIterator __first,
            _ForwardIterator __last)  [inline], [constexpr], [protected]
```

Memory expansion handler. Uses the member allocation function to obtain at least `n` objects worth of memory, copies `[first,last)` into it, reports what was actually allocated.
Referenced by reserve().

_M_range_check()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::_M_range_check (
            size_type __n) const  [inline], [constexpr], [protected]
```

Safety check used only from at().
Referenced by std::vector< _Tp, polymorphic_allocator< _Tp > >::at(), and std::vector< _Tp, polymorphic_allocator< _Tp > >::at().

assign() [1/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
void std::vector< _Tp, _Alloc >::assign (
            _InputIterator __first,
            _InputIterator __last)  [inline], [constexpr]
```

Assigns a range to a vector.

Parameters

__first	An input iterator.
__last	An input iterator.

This function fills a vector with copies of the elements in the range [__first,__last).
Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

assign() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
```

```
void std::vector< _Tp, _Alloc >::assign (
            initializer_list< value_type > __l)  [inline], [constexpr]
```

Assigns an initializer list to a vector.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a vector with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

assign() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::assign (
            size_type __n,
            const value_type & __val)  [inline], [constexpr]
```

Assigns a given value to a vector.

Parameters

__n	Number of elements to be assigned.
__val	Value to be assigned.

This function fills a vector with *__n* copies of the given value. Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

at() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::at (
            size_type __n)  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the vector.

Parameters

_↩ *_n*	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws out_of_range if the check fails.

at() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::at (
            size_type __n) const  [inline], [nodiscard], [constexpr]
```

Provides access to the data contained in the vector.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

Exceptions

std::out_of_range	If *__n* is an invalid index.

This function provides for safer data access. The parameter is first checked that it is in the range of the vector. The function throws out_of_range if the check fails.

back() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::back () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reference to the data at the last element of the vector.

back() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::back ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reference to the data at the last element of the vector.

begin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::begin () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

begin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::begin () [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Referenced by std::vector< _Tp, polymorphic_allocator< _Tp > >::vector(), std::vector< _Tp, polymorphic_allocator< _Tp > >::crend(), std::vector< _Tp, polymorphic_allocator< _Tp > >::empty(), std::vector< _Tp, polymorphic_allocator< _Tp > >::erase(), std::vector< _Tp, polymorphic_allocator< _Tp > >::erase(), std::vector< _Tp, polymorphic_allocator< _Tp > >::front(), std::vector< _Tp, polymorphic_allocator< _Tp > >::front(), std::vector< _Tp, polymorphic_allocator< _Tp > >::insert(), std::vector< _Tp, polymorphic_allocator< _Tp > >::insert(), std::vector< _Tp, polymorphic_allocator< _Tp > >::insert(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_parallel::multiway_merge_exact_splitting(), std::operator<=>(), operator=(), std::operator==(), std::vector< _Tp, polymorphic_allocator< _Tp > >::rend(), and std::vector< _Tp, polymorphic_allocator< _Tp > >::rend().

capacity()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::capacity () const [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns the total number of elements that the vector can hold before needing to allocate more memory.

Referenced by operator=(), and reserve().

cbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::cbegin () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points to the first element in the vector. Iteration is done in ordinary element order.

Referenced by std::vector< _Tp, polymorphic_allocator< _Tp > >::erase(), std::vector< _Tp, polymorphic_allocator< _Tp > >::erase(), std::vector< _Tp, polymorphic_allocator< _Tp > >::insert(), std::vector< _Tp, polymorphic_allocator< _Tp > >::insert(), and std::vector< _Tp, polymorphic_allocator< _Tp > >::insert().

cend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::cend () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

clear()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::clear () [inline], [constexpr], [noexcept]
```

Erases all the elements. Note that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

Referenced by operator=().

crbegin()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::crbegin () const [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

crend()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::crend () const  [inline], [constexpr], [noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

data()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
_Tp * std::vector< _Tp, _Alloc >::data ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a pointer such that [data(), data() + size()) is a valid range. For a non-empty vector, data() == &front().

emplace()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename... _Args>
iterator std::vector< _Tp, _Alloc >::emplace (
            const_iterator __position,
            _Args &&... __args)  [inline], [constexpr]
```

Inserts an object in vector before specified iterator.

Parameters

__position	A const_iterator into the vector.
__args	Arguments.

Returns

An iterator that points to the inserted data.

This function will insert an object of type T constructed with T(std::forward<Args>(args)...) before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

empty()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
bool std::vector< _Tp, _Alloc >::empty () const  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns true if the vector is empty. (Thus begin() would equal end().)
Referenced by std::discrete_distribution< _IntType >::probabilities().

end() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_iterator std::vector< _Tp, _Alloc >::end () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) iterator that points one past the last element in the vector. Iteration is done in ordinary element order.

end() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::end ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write iterator that points one past the last element in the vector. Iteration is done in ordinary element order.
Referenced by std::vector< _Tp, polymorphic_allocator< _Tp > >::vector(), std::vector< _Tp, polymorphic_allocator< _Tp > >::back(), std::vector< _Tp, polymorphic_allocator< _Tp > >::back(), std::vector< _Tp, polymorphic_allocator< _Tp > >::crbegin(), std::vector< _Tp, polymorphic_allocator< _Tp > >::empty(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_parallel::multiway_merge_exact_splitting(), std::operator<=>(), operator=(), std::operator==(), std::vector< _Tp, polymorphic_allo and std::vector< _Tp, polymorphic_allocator< _Tp > >::rbegin().

erase() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::erase (
            const_iterator __first,
            const_iterator __last)  [inline], [constexpr]
```

Remove a range of elements.

Parameters

__first	Iterator pointing to the first element to be erased.
__last	Iterator pointing to one past the last element to be erased.

Returns

An iterator pointing to the element pointed to by *__last* prior to erasing (or end()).

This function will erase the elements in the range [__first,__last) and shorten the vector accordingly.
Note This operation could be expensive and if it is frequently used the user should consider using std::list. The user is also cautioned that this function only erases the elements, and that if the elements themselves are pointers, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

erase() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::erase (
            const_iterator __position)  [inline], [constexpr]
```

Remove element at given position.

Parameters

__position	Iterator pointing to element to be erased.

Returns

An iterator pointing to the next element (or end()).

This function will erase the element at the given position and thus shorten the vector by one.
Note This operation could be expensive and if it is frequently used the user should consider using std::list. The user is also cautioned that this function only erases the element, and that if the element is itself a pointer, the pointed-to memory is not touched in any way. Managing the pointer is the user's responsibility.

front() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::front () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reference to the data at the first element of the vector.

front() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::front ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reference to the data at the first element of the vector.

get_allocator()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
allocator_type std::_Vector_base< _Tp, _Alloc >::get_allocator () const  [inline], [constexpr],
[noexcept]
```

Get a copy of the memory allocation object.
References get_allocator().
Referenced by get_allocator().

insert() [1/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
iterator std::vector< _Tp, _Alloc >::insert (
            const_iterator __position,
            _InputIterator __first,
            _InputIterator __last)  [inline], [constexpr]
```

Inserts a range into the vector.

Parameters

__position	A const_iterator into the vector.
__first	An input iterator.
__last	An input iterator.

Returns

An iterator that points to the inserted data.

This function will insert copies of the data in the range [__first,__last) into the vector before the location specified by *pos*. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

insert() [2/5]

```
template<typename _Tp, typename _Alloc>
vector< _Tp, _Alloc >::iterator vector::insert (
            const_iterator __position,
            const value_type & __x)  [constexpr]
```

Inserts given value into vector before specified iterator.

Parameters

__position	A const_iterator into the vector.

__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given value before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.
References vector(), std::begin(), std::cbegin(), std::end(), insert(), and std::move().
Referenced by insert().

insert() [3/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
            const_iterator __position,
            initializer_list< value_type > __l)  [inline], [constexpr]
```

Inserts an initializer_list into the vector.

Parameters

__position	An iterator into the vector.
__l	An initializer_list.

This function will insert copies of the data in the initializer_list *l* into the vector before the location specified by *position*. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

insert() [4/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
            const_iterator __position,
            size_type __n,
            const value_type & __x)  [inline], [constexpr]
```

Inserts a number of copies of given data into the vector.

Parameters

__position	A const_iterator into the vector.
__n	Number of elements to be inserted.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a specified number of copies of the given data before the location specified by *position*.
Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

insert() [5/5]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
iterator std::vector< _Tp, _Alloc >::insert (
            const_iterator __position,
            value_type && __x)  [inline], [constexpr]
```

Inserts given rvalue into vector before specified iterator.

Parameters

__position	A const_iterator into the vector.
__x	Data to be inserted.

Returns

An iterator that points to the inserted data.

This function will insert a copy of the given rvalue before the specified location. Note that this kind of operation could be expensive for a vector and if it is frequently used the user should consider using std::list.

max_size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::max_size () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns the size() of the largest possible vector.
Referenced by reserve().

operator=() [1/3]

```
template<typename _Tp, typename _Alloc>
vector< _Tp, _Alloc > & vector::operator= (
            const vector< _Tp, _Alloc > & __x)  [constexpr]
```

Vector assignment operator.

Parameters

_↩ _x	A vector of identical element and allocator types.

All the elements of *__x* are copied, but any unused capacity in *__x* will not be copied.
Whether the allocator is copied depends on the allocator traits.
References vector(), std::__addressof(), std::_Destroy(), begin(), capacity(), clear(), std::end(), end(), std::size(), and size().

operator=() [2/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
vector & std::vector< _Tp, _Alloc >::operator= (
            initializer_list< value_type > __l)  [inline], [constexpr]
```

Vector list assignment operator.

Parameters

↩ _↩ ↩ _↩ *l*	An initializer_list.

This function fills a vector with copies of the elements in the initializer list *__l*.
Note that the assignment completely changes the vector and that the resulting vector's size is the same as the number of elements assigned.

operator=() [3/3]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
vector & std::vector< _Tp, _Alloc >::operator= (
            vector< _Tp, _Alloc > && __x)  [inline], [constexpr], [noexcept]
```

Vector move assignment operator.

Parameters

_↩ *_x*	A vector of identical element and allocator types.

The contents of *__x* are moved into this vector (without copying, if the allocators permit it). Afterwards *__x* is a valid, but unspecified vector.
Whether the allocator is moved depends on the allocator traits.

operator[]() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reference std::vector< _Tp, _Alloc >::operator[] (
            size_type __n) const  [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the vector.

Parameters

_↩ *_n*	The index of the element for which data should be accessed.

Returns

Read-only (constant) reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

operator[]() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reference std::vector< _Tp, _Alloc >::operator[] (
            size_type __n)  [inline], [nodiscard], [constexpr], [noexcept]
```

Subscript access to the data contained in the vector.

Parameters

_↩ _n	The index of the element for which data should be accessed.

Returns

Read/write reference to data.

This operator allows for easy, array-style, data access. Note that data access with this operator is unchecked and out_of_range lookups are not defined. (For checked lookups see at().)

pop_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::pop_back ()  [inline], [constexpr], [noexcept]
```

Removes last element.
This is a typical stack operation. It shrinks the vector by one.
Note that no data is returned, and if the last element's data is needed, it should be retrieved before pop_back() is called.

push_back()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::push_back (
            const value_type & __x)  [inline], [constexpr]
```

Add data to the end of the vector.

Parameters

_↩ _x	Data to be added.

This is a typical stack operation. The function creates an element at the end of the vector and assigns the given data to it. Due to the nature of a vector this operation can be done in constant time if the vector has preallocated space available.
Referenced by __gnu_parallel::multiseq_partition(), and __gnu_parallel::multiseq_selection().

rbegin() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::rbegin () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

rbegin() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector< _Tp, _Alloc >::rbegin ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to the last element in the vector. Iteration is done in reverse element order.

rend() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
const_reverse_iterator std::vector< _Tp, _Alloc >::rend () const  [inline], [nodiscard], [constexpr],
[noexcept]
```

Returns a read-only (constant) reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

rend() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
reverse_iterator std::vector< _Tp, _Alloc >::rend ()  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns a read/write reverse iterator that points to one before the first element in the vector. Iteration is done in reverse element order.

reserve()

```
template<typename _Tp, typename _Alloc>
void vector::reserve (
            size_type __n)  [constexpr]
```

Attempt to preallocate enough memory for specified number of elements.

Parameters

_↩ _n	Number of elements required.

Exceptions

std::length_error	If *n* exceeds `max_size()`.

This function attempts to reserve enough memory for the vector to hold the specified number of elements. If the number requested is more than max_size(), length_error is thrown.
The advantage of this function is that if optimal code is a necessity and the user can determine the number of elements that will be required, the user can reserve the memory in advance, and thus prevent a possible reallocation of memory and copying of vector data.
References std::_Destroy(), _M_allocate_and_copy(), capacity(), max_size(), and std::size().

resize() [1/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::resize (
            size_type __new_size)  [inline], [constexpr]
```

Resizes the vector to the specified number of elements.

Parameters

__new_size	Number of elements the vector should contain.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise default constructed elements are appended.
Referenced by __gnu_parallel::__shrink_and_double(), and __gnu_parallel::multiway_merge_exact_splitting().

resize() [2/2]

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::resize (
            size_type __new_size,
            const value_type & __x)  [inline], [constexpr]
```

Resizes the vector to the specified number of elements.

Parameters

__new_size	Number of elements the vector should contain.
__x	Data with which new elements should be populated.

This function will resize the vector to the specified number of elements. If the number is smaller than the vector's current size the vector is truncated, otherwise the vector is extended and new elements are populated with given data.

shrink_to_fit()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::shrink_to_fit ()  [inline], [constexpr]
```

A non-binding request to reduce capacity() to size().

size()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
size_type std::vector< _Tp, _Alloc >::size () const  [inline], [nodiscard], [constexpr], [noexcept]
```

Returns the number of elements in the vector.
Referenced by __gnu_parallel::__shrink(), __gnu_parallel::__shrink_and_double(), std::vector< _Tp, polymorphic_allocator< _Tp > >::_
__gnu_parallel::list_partition(), operator=(), std::operator==(), std::vector< _Tp, polymorphic_allocator< _Tp > >::resize(),
and std::vector< _Tp, polymorphic_allocator< _Tp > >::resize().

swap()

```
template<typename _Tp, typename _Alloc = std::allocator<_Tp>>
void std::vector< _Tp, _Alloc >::swap (
            vector< _Tp, _Alloc > & __x)  [inline], [constexpr], [noexcept]
```

Swaps data with another vector.

Parameters

_↩ _x	A vector of the same element and allocator types.

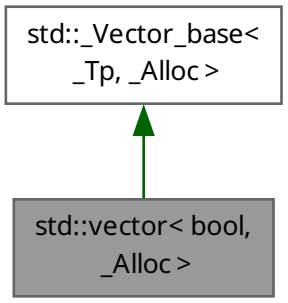
This exchanges the elements between two vectors in constant time. (Three pointers, so it should be quite fast.) Note that the global std::swap() function is specialized such that std::swap(v1,v2) will feed to this function.

Whether the allocators are swapped depends on the allocator traits.
The documentation for this class was generated from the following files:

- stl_vector.h
- vector.tcc

6.1039 std::vector< bool, _Alloc > Class Template Reference

`#include <vector>`
Inheritance diagram for std::vector< bool, _Alloc >:



Public Types

- typedef _Alloc **allocator_type**
- typedef _Bit_const_iterator **const_iterator**
- typedef const bool * **const_pointer**
- typedef bool **const_reference**
- typedef std::reverse_iterator< const_iterator > **const_reverse_iterator**
- typedef ptrdiff_t **difference_type**
- typedef _Bit_iterator **iterator**
- typedef _Bit_reference * **pointer**
- typedef _Bit_reference **reference**
- typedef std::reverse_iterator< iterator > **reverse_iterator**
- typedef size_t **size_type**
- typedef bool **value_type**

Public Member Functions

- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>>
  constexpr **vector** (_InputIterator __first, _InputIterator __last, const allocator_type &__a=allocator_type())
- constexpr **vector** (const allocator_type &__a) noexcept
- constexpr **vector** (const vector &__x)
- constexpr **vector** (const vector &__x, const __type_identity_t< allocator_type > &__a)
- constexpr **vector** (initializer_list< bool > __l, const allocator_type &__a=allocator_type())
- constexpr **vector** (size_type __n, const allocator_type &__a=allocator_type())
- constexpr **vector** (size_type __n, const bool &__value, const allocator_type &__a=allocator_type())

- **vector** (vector &&)=default
- constexpr **vector** (vector &&__x, const __type_identity_t< allocator_type > &__a) noexcept(_Bit_alloc_traits::↩ _S_always_equal())
- constexpr auto **_M_emplace_aux** (const_iterator __position, _Args &&... __args) -> iterator
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> constexpr void **assign** (_InputIterator __first, _InputIterator __last)
- constexpr void **assign** (initializer_list< bool > __l)
- constexpr void **assign** (size_type __n, const bool &__x)
- constexpr reference **at** (size_type __n)
- constexpr const_reference **at** (size_type __n) const
- constexpr reference **back** ()
- constexpr const_reference **back** () const
- constexpr const_iterator **begin** () const noexcept
- constexpr iterator **begin** () noexcept
- constexpr size_type **capacity** () const noexcept
- constexpr const_iterator **cbegin** () const noexcept
- constexpr const_iterator **cend** () const noexcept
- constexpr void **clear** () noexcept
- constexpr const_reverse_iterator **crbegin** () const noexcept
- constexpr const_reverse_iterator **crend** () const noexcept
- template<typename... _Args> constexpr iterator **emplace** (const_iterator __pos, _Args &&... __args)
- template<typename... _Args> constexpr reference **emplace_back** (_Args &&... __args)
- constexpr bool **empty** () const noexcept
- constexpr const_iterator **end** () const noexcept
- constexpr iterator **end** () noexcept
- constexpr iterator **erase** (const_iterator __first, const_iterator __last)
- constexpr iterator **erase** (const_iterator __position)
- constexpr void **flip** () noexcept
- constexpr reference **front** ()
- constexpr const_reference **front** () const
- constexpr allocator_type **get_allocator** () const
- constexpr allocator_type get_allocator () const noexcept
- constexpr iterator **insert** (const_iterator __p, initializer_list< bool > __l)
- iterator **insert** (const_iterator __position)
- template<typename _InputIterator, typename = std::_RequireInputIter<_InputIterator>> constexpr iterator **insert** (const_iterator __position, _InputIterator __first, _InputIterator __last)
- constexpr iterator **insert** (const_iterator __position, const bool &__x)
- constexpr iterator **insert** (const_iterator __position, size_type __n, const bool &__x)
- constexpr size_type **max_size** () const noexcept
- constexpr vector & **operator=** (const vector &__x)
- constexpr vector & **operator=** (initializer_list< bool > __l)
- constexpr vector & **operator=** (vector &&__x) noexcept(_Bit_alloc_traits::_S_nothrow_move())
- constexpr reference **operator[ ]** (size_type __n)
- constexpr const_reference **operator[ ]** (size_type __n) const
- constexpr void **pop_back** ()
- constexpr void **push_back** (bool __x)
- constexpr const_reverse_iterator **rbegin** () const noexcept
- constexpr reverse_iterator **rbegin** () noexcept
- constexpr const_reverse_iterator **rend** () const noexcept

- constexpr reverse_iterator **rend** () noexcept
- constexpr void **reserve** (size_type __n)
- constexpr void **resize** (size_type __new_size, bool __x=bool())
- constexpr void **shrink_to_fit** ()
- constexpr size_type **size** () const noexcept
- constexpr void **swap** (vector &__x) noexcept

Static Public Member Functions

- static constexpr size_t **_S_max_size** (const _Tp_alloc_type &__a) noexcept
- static constexpr void **swap** (reference __x, reference __y) noexcept

Protected Types

- typedef __gnu_cxx::__alloc_traits< _Alloc >::template rebind< _Bit_type >::other **_Bit_alloc_type**

Protected Member Functions

- constexpr pointer **_M_allocate** (size_t __n)
- constexpr _Alloc_result _M_allocate_and_copy (size_type __n, _ForwardIterator __first, _ForwardIterator __last)
- constexpr _Alloc_result **_M_allocate_at_least** (size_t __n)
- template<typename _ForwardIterator>
  constexpr void **_M_assign_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  constexpr void **_M_assign_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- constexpr void **_M_assign_dispatch** (_Integer __n, _Integer __val, __true_type)
- constexpr size_type **_M_check_len** (size_type __n, const char *__s) const
- constexpr iterator **_M_copy_aligned** (const_iterator __first, const_iterator __last, iterator __result)
- constexpr void **_M_create_storage** (size_t __n)
- constexpr void **_M_deallocate** ()
- constexpr void **_M_deallocate** (pointer __p, size_t __n)
- constexpr void **_M_deallocate** (pointer __p, size_t __n)
- constexpr void **_M_default_append** (size_type __n)
- constexpr void **_M_default_initialize** (size_type __n)
- constexpr iterator **_M_erase** (iterator __first, iterator __last)
- constexpr iterator **_M_erase** (iterator __pos)
- constexpr void **_M_erase_at_end** (iterator __pos)
- constexpr void **_M_fill_append** (size_type __n, const value_type &__x)
- constexpr void **_M_fill_assign** (size_t __n, bool __x)
- constexpr void **_M_fill_initialize** (size_type __n, const value_type &__value)
- constexpr void **_M_fill_insert** (iterator __position, size_type __n, bool __x)
- constexpr const _Bit_alloc_type & **_M_get_Bit_allocator** () const noexcept
- constexpr _Bit_alloc_type & **_M_get_Bit_allocator** () noexcept
- constexpr const _Tp_alloc_type & **_M_get_Tp_allocator** () const noexcept
- constexpr _Tp_alloc_type & **_M_get_Tp_allocator** () noexcept
- constexpr void **_M_initialize** (size_type __n)
- template<typename _ForwardIterator>
  constexpr void **_M_initialize_range** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator>
  constexpr void **_M_initialize_range** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- constexpr void **_M_initialize_value** (bool __x) noexcept
- constexpr void **_M_insert_aux** (iterator __position, bool __x)

- constexpr void **_M_insert_dispatch** (iterator __pos, _Integer __n, _Integer __val, __true_type)
- template<typename _InputIterator>
  constexpr void **_M_insert_range** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _ForwardIterator>
  constexpr void **_M_insert_range** (iterator __position, _ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- constexpr iterator **_M_insert_rval** (const_iterator __position, value_type &&__v)
- constexpr void **_M_move_data** (_Bvector_base &&__x) noexcept
- constexpr void **_M_range_check** (size_type __n) const
- constexpr void **_M_range_initialize** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- constexpr void **_M_range_initialize_n** (_Iterator __first, _Sentinel __last, size_type __n)
- constexpr void **_M_range_insert** (iterator __pos, _InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- constexpr void **_M_realloc_append** (_Args &&... __args)
- constexpr void **_M_realloc_insert** (iterator __position, _Args &&... __args)
- constexpr void **_M_reallocate** (size_type __n)
- constexpr void **_M_replace_storage** (pointer __start, pointer __end, size_type __cap)
- constexpr void **_M_replace_with** (size_type __n, _ForwardIterator __first, _ForwardIterator __last)
- constexpr bool **_M_shrink_to_fit** ()
- void **data** ()=delete

Static Protected Member Functions

- static constexpr size_type **_S_check_init_len** (size_type __n, const allocator_type &__a)
- static constexpr size_t **_S_nword** (size_t __n)

Protected Attributes

- _Vector_impl **_M_impl**

Friends

- struct **std::hash**< **vector** >

6.1039.1 Detailed Description

template<**typename _Alloc**>
class std::vector< **bool, _Alloc** >

A specialization of vector for booleans which offers fixed time access to individual elements in any order.

Since

C++98

Template Parameters

_Alloc	Allocator type.

Note that vector<bool> does not actually meet the requirements for being a container. This is because the reference and pointer types are not really references and pointers to bool. See DR96 for details.

See also

vector for function documentation.

In some terminology a vector can be described as a dynamic C-style array, it offers fast and efficient access to individual elements in any order and saves the user from worrying about memory and size allocation. Subscripting ([]) access is also provided as with C-style arrays.

6.1039.2 Member Function Documentation

_M_allocate_and_copy()

```
_Alloc_result std::vector< bool, _Alloc >::_M_allocate_and_copy (
            size_type __n,
            _ForwardIterator __first,
            _ForwardIterator __last)  [inline], [constexpr], [protected]
```

Memory expansion handler. Uses the member allocation function to obtain at least `n` objects worth of memory, copies `[first,last)` into it, reports what was actually allocated.

get_allocator()

```
() const
```

Get a copy of the memory allocation object.
The documentation for this class was generated from the following files:

- stl_bvector.h
- vector.tcc

6.1040 std::ranges::view_base Struct Reference

`#include <ranges_base.h>`

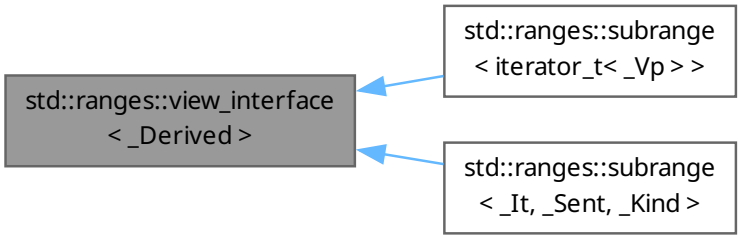
6.1040.1 Detailed Description

[range.view] The ranges::view_base type.
The documentation for this struct was generated from the following file:

- ranges_base.h

6.1041 std::ranges::view_interface< _Derived > Class Template Reference

`#include <ranges_util.h>`
Inheritance diagram for std::ranges::view_interface< _Derived >:



Public Member Functions

- constexpr decltype(auto) **back** ()
- constexpr decltype(auto) **back** () const
- constexpr auto **cbegin** ()

- constexpr auto **cbegin** () const
- constexpr auto **cend** ()
- constexpr auto **cend** () const
- constexpr auto **data** () const noexcept(noexcept(ranges::begin(_M_derived())))
- constexpr auto **data** () noexcept(noexcept(ranges::begin(_M_derived())))
- constexpr bool **empty** () const noexcept(noexcept(_S_empty(_M_derived())))
- constexpr bool **empty** () const noexcept(noexcept(ranges::size(_M_derived())==0))
- constexpr bool **empty** () noexcept(noexcept(_S_empty(_M_derived())))
- constexpr bool **empty** () noexcept(noexcept(ranges::size(_M_derived())==0))
- constexpr decltype(auto) **front** ()
- constexpr decltype(auto) **front** () const
- constexpr **operator bool** () const noexcept(noexcept(ranges::empty(_M_derived())))
- constexpr **operator bool** () noexcept(noexcept(ranges::empty(_M_derived())))
- template<random_access_range _Range = _Derived>
  constexpr decltype(auto) **operator[ ]** (range_difference_t< _Range > __n)
- template<random_access_range _Range = const _Derived>
  constexpr decltype(auto) **operator[ ]** (range_difference_t< _Range > __n) const
- constexpr auto **size** () const noexcept(noexcept(_S_size(_M_derived())))
- constexpr auto **size** () noexcept(noexcept(_S_size(_M_derived())))

6.1041.1 Detailed Description

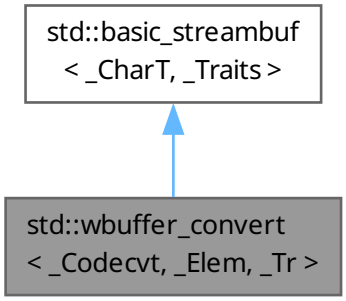
template<typename _Derived>
requires is_class_v<_Derived> && same_as<_Derived, remove_cv_t<_Derived>>
class std::ranges::view_interface< _Derived >

The ranges::view_interface class template.
The documentation for this class was generated from the following files:

- ranges_base.h
- ranges_util.h

6.1042 std::wbuffer_convert< _Codecvt, _Elem, _Tr > Class Template Reference

`#include <locale_conv.h>`
Inheritance diagram for std::wbuffer_convert< _Codecvt, _Elem, _Tr >:



Public Types

- typedef _Codecvt::state_type **state_type**

- typedef _CharT char_type
- typedef _Traits traits_type
- typedef traits_type::int_type int_type
- typedef traits_type::pos_type pos_type
- typedef traits_type::off_type off_type

- typedef basic_streambuf< char_type, traits_type > __streambuf_type

Public Member Functions

- wbuffer_convert ()
- **wbuffer_convert** (const wbuffer_convert &)=delete
- wbuffer_convert (streambuf *__bytebuf, _Codecvt *__pcvt=new _Codecvt, state_type __state=state_type())
- locale getloc () const
- streamsize in_avail ()
- wbuffer_convert & **operator=** (const wbuffer_convert &)=delete
- locale pubimbue (const locale &__loc)
- streambuf * **rdbuf** () const noexcept
- streambuf * **rdbuf** (streambuf *__bytebuf) noexcept
- int_type sbumpc ()
- int_type sgetc ()
- streamsize sgetn (char_type *__s, streamsize __n)
- int_type snextc ()
- int_type sputbackc (char_type __c)
- int_type sputc (char_type __c)
- streamsize sputn (const char_type *__s, streamsize __n)
- state_type state () const noexcept
- int_type sungetc ()

- basic_streambuf * pubsetbuf (char_type *__s, streamsize __n)
- pos_type pubseekoff (off_type __off, ios_base::seekdir __way, ios_base::openmode __mode=ios_base::in|ios_base::out)
- pos_type pubseekpos (pos_type __sp, ios_base::openmode __mode=ios_base::in|ios_base::out)
- int pubsync ()

Protected Member Functions

- void **__safe_gbump** (streamsize __n)
- void **__safe_pbump** (streamsize __n)
- void gbump (int __n)
- virtual void imbue (const locale &__loc)
- _Wide_streambuf::int_type overflow (typename _Wide_streambuf::int_type __out)
- virtual int_type pbackfail (int_type __c=traits_type::eof())
- void pbump (int __n)
- virtual pos_type seekoff (off_type, ios_base::seekdir, ios_base::openmode=ios_base::in|ios_base::out)
- virtual pos_type seekpos (pos_type, ios_base::openmode=ios_base::in|ios_base::out)
- virtual basic_streambuf< char_type, _Traits > * setbuf (char_type *, streamsize)

- void setg (char_type *__gbeg, char_type *__gnext, char_type *__gend)
- void setp (char_type *__pbeg, char_type *__pend)
- virtual streamsize showmanyc ()
- void **swap** (basic_streambuf &__sb)
- int sync ()
- virtual int_type uflow ()
- _Wide_streambuf::int_type underflow ()
- virtual streamsize xsgetn (char_type *__s, streamsize __n)
- virtual streamsize xsputn (const char_type *__s, streamsize __n)
- streamsize **xsputn** (const typename _Wide_streambuf::char_type *__s, streamsize __n)

- char_type * eback () const
- char_type * gptr () const
- char_type * egptr () const

- char_type * pbase () const
- char_type * pptr () const
- char_type * epptr () const

Protected Attributes

- locale _M_buf_locale
- char_type * _M_in_beg
- char_type * _M_in_cur
- char_type * _M_in_end
- char_type * _M_out_beg
- char_type * _M_out_cur
- char_type * _M_out_end

6.1042.1 Detailed Description

template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
class std::wbuffer_convert< _Codecvt, _Elem, _Tr >

Buffer conversions.

6.1042.2 Member Typedef Documentation

__streambuf_type

```
template<typename _CharT, typename _Traits>
typedef basic_streambuf<char_type, traits_type> std::basic_streambuf< _CharT, _Traits >::__↩
streambuf_type  [inherited]
```

This is a non-standard type.

char_type

```
template<typename _CharT, typename _Traits>
typedef _CharT std::basic_streambuf< _CharT, _Traits >::char_type  [inherited]
```

These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

int_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::int_type std::basic_streambuf< _CharT, _Traits >::int_type  [inherited]
```
These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

off_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::off_type std::basic_streambuf< _CharT, _Traits >::off_type  [inherited]
```
These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

pos_type

```
template<typename _CharT, typename _Traits>
typedef traits_type::pos_type std::basic_streambuf< _CharT, _Traits >::pos_type  [inherited]
```
These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

traits_type

```
template<typename _CharT, typename _Traits>
typedef _Traits std::basic_streambuf< _CharT, _Traits >::traits_type  [inherited]
```
These are standard types. They permit a standardized way of referring to names of (or names dependent on) the template parameters, which are specific to the implementation.

6.1042.3 Constructor & Destructor Documentation

wbuffer_convert() [1/2]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert ()  [inline]
```
Default constructor.
References wbuffer_convert().
Referenced by wbuffer_convert().

wbuffer_convert() [2/2]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert (
            streambuf * __bytebuf,
            _Codecvt * __pcvt = new _Codecvt,
            state_type __state = state_type())  [inline], [explicit]
```
Constructor.

Parameters

__bytebuf	The underlying byte stream buffer.
__pcvt	The facet to use for conversions.
__state	Initial conversion state.

Takes ownership of __pcvt and will delete it in the destructor.
References std::basic_streambuf< _CharT, _Traits >::setg(), and std::basic_streambuf< _CharT, _Traits >::setp().

6.1042.4 Member Function Documentation

eback()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::eback () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _Cha std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_filebuf< _CharT, _Traits >::underflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

egptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::egptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_streambuf< char_type, traits_type >::in_avail std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sgetc(), std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< char std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(), std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and xsgetn().

epptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::epptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), std::basic_streambuf< char_type, traits_type >::sputc(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

gbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::gbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the read position.

Parameters

_↩ _n	The delta by which to move.

This just advances the read position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< char_type, traits_type >::sbumpc(), std::basic_streambuf< char_type, traits_type >::sputbackc(), std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_streambuf< char_type, traits_type >::uflow(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

getloc()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::getloc () const  [inline], [inherited]
```

Locale access.

Returns

The current locale in effect.

If pubimbue(loc) has been called, then the most recent `loc` is returned. Otherwise the global locale in effect at the time of construction is returned.
Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

gptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::gptr () const  [inline], [protected], [inherited]
```

Access to the get area.
These functions are only available to other protected functions, including derived classes.

- eback() returns the beginning pointer for the input sequence
- gptr() returns the next pointer for the input sequence
- egptr() returns the end pointer for the input sequence

Referenced by std::basic_filebuf< char_type, traits_type >::_M_create_pback(), std::basic_filebuf< char_type, traits_type >::_M_destroy
std::basic_filebuf< _CharT, _Traits >::imbue(), std::basic_streambuf< char_type, traits_type >::in_avail(), std::basic_stringbuf< _CharT,
std::basic_filebuf< _CharT, _Traits >::pbackfail(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::pbackfail(), std::basic_streambuf< cha
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_streambuf< char_type, traits_type >::sgetc(),
std::basic_filebuf< _CharT, _Traits >::showmanyc(), std::basic_stringbuf< char >::showmanyc(), std::basic_streambuf< char_type, traits
std::basic_streambuf< char_type, traits_type >::sungetc(), std::basic_streambuf< char_type, traits_type >::uflow(),
std::basic_filebuf< _CharT, _Traits >::underflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::underflow(),
std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow(), std::basic_filebuf< _CharT, _Traits >::xsgetn(), and
xsgetn().

imbue()

```
template<typename _CharT, typename _Traits>
virtual void std::basic_streambuf< _CharT, _Traits >::imbue (
            const locale & __loc)  [inline], [protected], [virtual], [inherited]
```

Changes translations.

Parameters

__loc	A new locale.

Translations done during I/O which depend on the current locale are changed by this call. The standard adds, *Between invocations of this function a class derived from streambuf can safely cache results of calls to locale functions and to members of facets so obtained.*

Note

Base class version does nothing.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, and std::basic_filebuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubimbue().

in_avail()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::in_avail ()  [inline], [inherited]
```

Looking ahead into the stream.

Returns

The number of characters available.

If a read position is available, returns the number of characters available for reading before the buffer must be refilled. Otherwise returns the derived `showmanyc()`.

overflow()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
_Wide_streambuf::int_type std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow (
            typename _Wide_streambuf::int_type __c)  [inline], [protected], [virtual]
```

Consumes data from the buffer; writes to the controlled sequence.

Parameters

__c	An additional character to consume.

Returns

eof() to indicate failure, something else (usually *__c*, or not_eof())

Informally, this function is called when the output buffer is full (or does not exist, as buffering need not actually be done). If a buffer exists, it is *consumed*, with *some effect* on the controlled sequence. (Typically, the buffer is written out to the sequence verbatim.) In either case, the character *c* is also written out, if *__c* is not `eof()`.
For a formal definition of this function, see a good text such as Langer & Kreft, or [27.5.2.4.5]/3-7.
A functioning output streambuf can be created by overriding only this function (no buffer area will be used).

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References std::basic_streambuf< _CharT, _Traits >::sputc().

pbackfail()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::pbackfail (
            int_type __c = traits_type::eof())  [inline], [protected], [virtual], [inherited]
```

Tries to back up the input sequence.

Parameters

_↩ _c	The character to be inserted back into the sequence.

Returns

eof() on failure, *some other value* on success

Postcondition

The constraints of `gptr()`, `eback()`, and `pptr()` are the same as for `underflow()`.

Note

Base class version does nothing, returns eof().

Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Tr std::basic_stringbuf< char >, and std::basic_stringbuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::sputbackc(), and std::basic_streambuf< char_type, traits_type >::sunget

pbase()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pbase () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_stringbuf< _CharT, std::basic_stringbuf< char >::str(), std::basic_filebuf< _CharT, _Traits >::sync(), and std::basic_filebuf< _CharT, _Traits >::xsputn().

pbump()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::pbump (
            int __n)  [inline], [protected], [inherited]
```

Moving the write position.

Parameters

_↩ _n	The delta by which to move.

This just advances the write position without returning any data.
Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), and std::basic_streambuf< char_type, traits_type >::sputc().

pptr()

```
template<typename _CharT, typename _Traits>
char_type * std::basic_streambuf< _CharT, _Traits >::pptr () const  [inline], [protected], [inherited]
```

Access to the put area.
These functions are only available to other protected functions, including derived classes.

- pbase() returns the beginning pointer for the output sequence
- pptr() returns the next pointer for the output sequence
- epptr() returns the end pointer for the output sequence

Referenced by std::basic_filebuf< _CharT, _Traits >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_filebuf< _CharT, _Traits >::seekoff(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(), std::basic_streambuf< char_t std::basic_filebuf< _CharT, _Traits >::sync(), std::basic_filebuf< _CharT, _Traits >::xsputn(), and xsputn().

pubimbue()

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::pubimbue (
            const locale & __loc)  [inline], [inherited]
```

Entry point for imbue().

Parameters

__loc	The new locale.

Returns

The previous locale.

Calls the derived imbue(__loc).

pubseekoff()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekoff (
            off_type __off,
```

```
            ios_base::seekdir __way,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__off	Offset.
__way	Value for ios_base::seekdir.
__mode	Value for ios_base::openmode.

Calls virtual seekoff function.

pubseekpos()

```
template<typename _CharT, typename _Traits>
pos_type std::basic_streambuf< _CharT, _Traits >::pubseekpos (
            pos_type __sp,
            ios_base::openmode __mode = ios_base::in | ios_base::out)  [inline], [inherited]
```

Alters the stream position.

Parameters

__sp	Position
__mode	Value for ios_base::openmode.

Calls virtual seekpos function.

pubsetbuf()

```
template<typename _CharT, typename _Traits>
basic_streambuf * std::basic_streambuf< _CharT, _Traits >::pubsetbuf (
            char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry points for derived buffer functions.
The public versions of `pubfoo` dispatch to the protected derived `foo` member functions, passing the arguments (if any) and returning the result unchanged.

pubsync()

```
template<typename _CharT, typename _Traits>
int std::basic_streambuf< _CharT, _Traits >::pubsync ()  [inline], [inherited]
```

Calls virtual sync function.
Referenced by std::basic_istream< _CharT, _Traits >::sync().

sbumpc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sbumpc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character and increments the read pointer, otherwise calls and returns `uflow()`.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::ignore(), and std::basic_streambuf< char_type, traits_type >::snextc().

seekoff()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekoff (
            off_type ,
            ios_base::seekdir ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Ti std::basic_stringbuf< char >, and std::basic_stringbuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubseekoff().

seekpos()

```
template<typename _CharT, typename _Traits>
virtual pos_type std::basic_streambuf< _CharT, _Traits >::seekpos (
            pos_type ,
            ios_base::openmode  = ios_base::in | ios_base::out)  [inline], [protected], [virtual],
[inherited]
```

Alters the stream positions.
Each derived class provides its own appropriate behavior.

Note

Base class version does nothing, returns a `pos_type` that represents an invalid stream position.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Ti std::basic_stringbuf< char >, and std::basic_stringbuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubseekpos().

setbuf()

```
template<typename _CharT, typename _Traits>
virtual basic_streambuf< char_type, _Traits > * std::basic_streambuf< _CharT, _Traits >::setbuf
(
            char_type * ,
            streamsize )  [inline], [protected], [virtual], [inherited]
```

Manipulates the buffer.
Each derived class provides its own appropriate behavior. See the next-to-last paragraph of https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html#io.streambuf.buffering for more on this function.

Note

Base class version does nothing, returns `this`.

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, and std::basic_stringbuf< char >.
Referenced by std::basic_streambuf< char_type, traits_type >::pubsetbuf().

setg()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setg (
             char_type * __gbeg,
             char_type * __gnext,
             char_type * __gend)  [inline], [protected], [inherited]
```

Setting the three read area pointers.

Parameters

__gbeg	A pointer.
__gnext	A pointer.
__gend	A pointer.

Postcondition

__gbeg == `eback()`, *__gnext* == `gptr()`, and *__gend* == `egptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), std::basic_filebuf< char_type, traits_type >::_M_create
std::basic_filebuf< char_type, traits_type >::_M_destroy_pback(), std::basic_filebuf< char_type, traits_type >::_M_set_buffer(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::overflow(), std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekoff(),
std::basic_stringbuf< _CharT, _Traits, _Alloc >::seekpos(), and std::basic_filebuf< _CharT, _Traits >::xsgetn().

setp()

```
template<typename _CharT, typename _Traits>
void std::basic_streambuf< _CharT, _Traits >::setp (
             char_type * __pbeg,
             char_type * __pend)  [inline], [protected], [inherited]
```

Setting the three write area pointers.

Parameters

__pbeg	A pointer.
__pend	A pointer.

Postcondition

__pbeg == `pbase()`, *__pbeg* == `pptr()`, and *__pend* == `epptr()`

Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::wbuffer_convert(), and std::basic_filebuf< char_type, traits_type >::_M_s

sgetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sgetc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

If the input read position is available, returns that character, otherwise calls and returns `underflow()`. Does not move the read position after fetching the character.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, _Traits >::ignore(), std::basic_istream< _CharT, _Traits >::igr and std::basic_streambuf< char_type, traits_type >::snextc().

sgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sgetn (
             char_type * __s,
             streamsize __n)  [inline], [inherited]
```

Entry point for xsgetn.

Parameters

_↩ _*s*	A buffer area.
_↩ _*n*	A count.

Returns xsgetn(__s,__n). The effect is to fill __*s*[0] through __*s*[__n-1] with characters from the input sequence, if possible.

showmanyc()

```
template<typename _CharT, typename _Traits>
virtual streamsize std::basic_streambuf< _CharT, _Traits >::showmanyc ()  [inline], [protected],
[virtual], [inherited]
```

Investigating the data available.

Returns

An estimate of the number of characters available in the input sequence, or -1.

If it returns a positive value, then successive calls to `underflow()` *will not return* `traits::eof()` *until at least that number of characters have been supplied. If* `showmanyc()` *returns -1, then calls to* `underflow()` *or* `uflow()` *will fail.* [27.5.2.4.3]/1

Note

Base class version does nothing, returns zero.

The standard adds that *the intention is not only that the calls [to underflow or uflow] will not return* `eof()` *but that they will return immediately.*

The standard adds that *the morphemes of* `showmanyc` *are* ***es-how-many-see****, not* ***show-manic****.*

Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, std::basic_filebuf< char_type, traits_type >, std::basic_filebuf< wchar_t >, std::basic_stringbuf< _CharT, _Traits, _Alloc >, std::basic_stringbuf< char >, and std::basic_stringbuf< wchar_t >.
Referenced by std::basic_streambuf< char_type, traits_type >::in_avail().

snextc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::snextc ()  [inline], [inherited]
```

Getting the next character.

Returns

The next character, or eof.

Calls `sbumpc()`, and if that function returns `traits::eof()`, so does this function. Otherwise, `sgetc()`.
Referenced by std::basic_iostream< char_type, traits_type >::get(), std::basic_istream< _CharT, _Traits >::get(), std::basic_istream< _CharT, char_traits< _CharT > >::get(), and std::basic_istream< _CharT, char_traits< _CharT > >::readsome().

sputbackc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputbackc (
            char_type __c)  [inline], [inherited]
```

Pushing characters back into the input stream.

Parameters

_↩ _c	The character to push back.

Returns

The previous character, if possible.

Similar to sungetc(), but __c is pushed onto the stream instead of *the previous character.* If successful, the next character fetched from the input stream will be __c.
Referenced by std::basic_istream< _CharT, _Traits >::putback().

sputc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sputc (
            char_type __c)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _c	A character to output.

Returns

__*c*, if possible.

One of two public output functions.
If a write position is available for the output sequence (i.e., the buffer is not full), stores __*c* in that position, increments the position, and returns `traits::to_int_type(__c)`. If a write position is not available, returns `overflow(↩ __c)`.
Referenced by std::wbuffer_convert< _Codecvt, _Elem, _Tr >::overflow().

sputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::sputn (
            const char_type * __s,
            streamsize __n)  [inline], [inherited]
```

Entry point for all single-character output functions.

Parameters

_↩ _*s*	A buffer read area.
_↩ _*n*	A count.

One of two public output functions.
Returns xsputn(__s,__n). The effect is to write __*s*[0] through __*s*[__n-1] to the output sequence, if possible.

state()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
state_type std::wbuffer_convert< _Codecvt, _Elem, _Tr >::state () const  [inline], [noexcept]
```

The conversion state following the last conversion.

sungetc()

```
template<typename _CharT, typename _Traits>
int_type std::basic_streambuf< _CharT, _Traits >::sungetc ()  [inline], [inherited]
```

Moving backwards in the input stream.

Returns

The previous character, if possible.

If a putback position is available, this function decrements the input pointer and returns that character. Otherwise, calls and returns pbackfail(). The effect is to *unget* the last character *gotten*.
Referenced by std::basic_istream< _CharT, _Traits >::unget().

sync()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
int std::wbuffer_convert< _Codecvt, _Elem, _Tr >::sync ()  [inline], [protected], [virtual]
```

Synchronizes the buffer arrays with the controlled sequences.

Returns

-1 on failure.

Each derived class provides its own appropriate behavior, including the definition of *failure*.

Note

Base class version does nothing, returns zero.

Reimplemented from std::basic_streambuf< _CharT, _Traits >.

uflow()

```
template<typename _CharT, typename _Traits>
virtual int_type std::basic_streambuf< _CharT, _Traits >::uflow ()  [inline], [protected], [virtual],
[inherited]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function does the same thing as `underflow()`, and in fact is required to call that function. It also returns the new character, like `underflow()` does. However, this function also moves the read position forward by one.
Reimplemented in __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >.
Referenced by std::basic_streambuf< char_type, traits_type >::sbumpc(), and xsgetn().

underflow()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Tr = char_traits<_Elem>>
_Wide_streambuf::int_type std::wbuffer_convert< _Codecvt, _Elem, _Tr >::underflow ()  [inline],
[protected], [virtual]
```

Fetches more data from the controlled sequence.

Returns

The first character from the *pending sequence*.

Informally, this function is called when the input buffer is exhausted (or does not exist, as buffering need not actually be done). If a buffer exists, it is *refilled*. In either case, the next available character is returned, or `traits::eof()` to indicate a null pending sequence.
For a formal definition of the pending sequence, see a good text such as Langer & Kreft, or [27.5.2.4.3]/7-14.
A functioning input streambuf can be created by overriding only this function (no buffer area will be used). For an example, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/streambufs.html

Note

Base class version does nothing, returns eof().

Reimplemented from std::basic_streambuf< _CharT, _Traits >.
References std::basic_streambuf< _CharT, _Traits >::egptr(), and std::basic_streambuf< _CharT, _Traits >::gptr().

xsgetn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsgetn (
            char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character extraction.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to assign.

Returns

The number of characters assigned.

Fills __s[0] through __s[__n-1] with characters from the input sequence, as if by `sbumpc()`. Stops when either __n characters have been copied, or when `traits::eof()` would be copied.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, and std::basic_filebuf< char_type, traits_type >.
References egptr(), gptr(), std::min(), and uflow().
Referenced by std::basic_streambuf< char_type, traits_type >::sgetn().

xsputn()

```
template<typename _CharT, typename _Traits>
streamsize std::basic_streambuf< _CharT, _Traits >::xsputn (
            const char_type * __s,
            streamsize __n)  [protected], [virtual], [inherited]
```

Multiple character insertion.

Parameters

_↩ _s	A buffer area.
_↩ _n	Maximum number of characters to write.

Returns

The number of characters written.

Writes __s[0] through __s[__n-1] to the output sequence, as if by `sputc()`. Stops when either *n* characters have been copied, or when `sputc()` would return `traits::eof()`.
It is expected that derived classes provide a more efficient implementation by overriding this definition.
Reimplemented in std::basic_filebuf< _CharT, _Traits >, std::basic_filebuf< _CharT, encoding_char_traits< _CharT > >, std::basic_filebuf< _CharT, std::char_traits< _CharT > >, std::basic_filebuf< char >, and std::basic_filebuf< char_type, traits_type >.
References epptr(), std::min(), overflow(), and pptr().
Referenced by std::basic_streambuf< char_type, traits_type >::sputn().

6.1042.5 Member Data Documentation

_M_buf_locale

```
template<typename _CharT, typename _Traits>
locale std::basic_streambuf< _CharT, _Traits >::_M_buf_locale  [protected], [inherited]
```

Current locale setting.
Referenced by std::basic_filebuf< _CharT, _Traits >::basic_filebuf().

_M_in_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_beg  [protected], [inherited]
```
Start of get area.

_M_in_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_cur  [protected], [inherited]
```
Current read area.

_M_in_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_in_end  [protected], [inherited]
```
End of get area.

_M_out_beg

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_beg  [protected], [inherited]
```
Start of put area.

_M_out_cur

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_cur  [protected], [inherited]
```
Current put area.

_M_out_end

```
template<typename _CharT, typename _Traits>
char_type* std::basic_streambuf< _CharT, _Traits >::_M_out_end  [protected], [inherited]
```
End of put area.
The documentation for this class was generated from the following file:

- locale_conv.h

6.1043 std::weak_ptr< _Tp > Class Template Reference

```
#include <memory>
```

Public Types

- using **element_type**

Public Member Functions

- template<typename _Yp, typename = _Constructible<const shared_ptr<_Yp>&>>
  weak_ptr (const shared_ptr< _Yp > &__r) noexcept
- **weak_ptr** (const weak_ptr &) noexcept=default

- template<typename _Yp, typename = _Constructible<const weak_ptr<_Yp>&>>
  weak_ptr (const weak_ptr< _Yp > &__r) noexcept
- **weak_ptr** (weak_ptr &&) noexcept=default
- template<typename _Yp, typename = _Constructible<weak_ptr<_Yp>>>
  weak_ptr (weak_ptr< _Yp > &&__r) noexcept
- bool **expired** () const noexcept
- shared_ptr< _Tp > **lock** () const noexcept
- template<typename _Yp>
  _Assignable< const shared_ptr< _Yp > & > **operator=** (const shared_ptr< _Yp > &__r) noexcept
- weak_ptr & **operator=** (const weak_ptr &__r) noexcept=default
- template<typename _Yp>
  _Assignable< const weak_ptr< _Yp > & > **operator=** (const weak_ptr< _Yp > &__r) noexcept
- weak_ptr & **operator=** (weak_ptr &&__r) noexcept=default
- template<typename _Yp>
  _Assignable< weak_ptr< _Yp > > **operator=** (weak_ptr< _Yp > &&__r) noexcept
- template<typename _Tp1>
  bool **owner_before** (const __shared_ptr< _Tp1, _Lp > &__rhs) const noexcept
- template<typename _Tp1>
  bool **owner_before** (const __weak_ptr< _Tp1, _Lp > &__rhs) const noexcept
- void **reset** () noexcept
- void **swap** (__weak_ptr &__s) noexcept
- long **use_count** () const noexcept

Related Symbols

(Note that these are not member symbols.)

- template<typename _Tp>
  void swap (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept

6.1043.1 Detailed Description

template<typename _Tp>
class std::weak_ptr< _Tp >

A non-owning observer for a pointer owned by a shared_ptr.

Since

C++11

A weak_ptr provides a safe alternative to a raw pointer when you want a non-owning reference to an object that is managed by a shared_ptr.
Unlike a raw pointer, a weak_ptr can be converted to a new shared_ptr that shares ownership with every other shared_ptr that already owns the pointer. In other words you can upgrade from a non-owning "weak" reference to an owning shared_ptr, without having access to any of the existing shared_ptr objects.
Also unlike a raw pointer, a weak_ptr does not become "dangling" after the object it points to has been destroyed. Instead, a weak_ptr becomes *expired* and can no longer be converted to a shared_ptr that owns the freed pointer, so you cannot accidentally access the pointed-to object after it has been destroyed.
The documentation for this class was generated from the following file:

- bits/shared_ptr.h

6.1044 std::weibull_distribution< _RealType > Class Template Reference

`#include <random>`

Classes

- struct param_type

Public Types

- typedef _RealType result_type

Public Member Functions

- **weibull_distribution** (_RealType __a, _RealType __b=_RealType(1))
- **weibull_distribution** (const param_type &__p)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng)
- template<typename _ForwardIterator, typename _UniformRandomNumberGenerator>
  void **__generate** (_ForwardIterator __f, _ForwardIterator __t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- template<typename _UniformRandomNumberGenerator>
  void **__generate** (result_type *__f, result_type *__t, _UniformRandomNumberGenerator &__urng, const param_type &__p)
- _RealType a () const
- _RealType b () const
- result_type max () const
- result_type min () const
- template<typename _UniformRandomNumberGenerator>
  result_type operator() (_UniformRandomNumberGenerator &__urng)
- template<typename _UniformRandomNumberGenerator>
  result_type **operator()** (_UniformRandomNumberGenerator &__urng, const param_type &__p)
- param_type param () const
- void param (const param_type &__param)
- void reset ()

Friends

- bool operator== (const weibull_distribution &__d1, const weibull_distribution &__d2)

6.1044.1 Detailed Description

template<typename _RealType = double>
class std::weibull_distribution< _RealType >

A weibull_distribution random number distribution.
The formula for the normal probability density function is:

$$p(x|\alpha,\beta) = \frac{\alpha}{\beta}(\frac{x}{\beta})^{\alpha-1}\exp{(-(\frac{x}{\beta})^{\alpha})}$$

Since

C++11

6.1044.2 Member Typedef Documentation

result_type

```
template<typename _RealType = double>
typedef _RealType std::weibull_distribution< _RealType >::result_type
```

The type of the range of the distribution.

6.1044.3 Member Function Documentation

a()

```
template<typename _RealType = double>
_RealType std::weibull_distribution< _RealType >::a () const  [inline]
```

Return the a parameter of the distribution.

b()

```
template<typename _RealType = double>
_RealType std::weibull_distribution< _RealType >::b () const  [inline]
```

Return the b parameter of the distribution.

max()

```
template<typename _RealType = double>
result_type std::weibull_distribution< _RealType >::max () const  [inline]
```

Returns the least upper bound value of the distribution.

min()

```
template<typename _RealType = double>
result_type std::weibull_distribution< _RealType >::min () const  [inline]
```

Returns the greatest lower bound value of the distribution.

operator()()

```
template<typename _RealType = double>
template<typename _UniformRandomNumberGenerator>
result_type std::weibull_distribution< _RealType >::operator() (
            _UniformRandomNumberGenerator & __urng)  [inline]
```

Generating functions.
Referenced by operator()().

param() [1/2]

```
template<typename _RealType = double>
param_type std::weibull_distribution< _RealType >::param () const  [inline]
```

Returns the parameter set of the distribution.
Referenced by std::operator>>().

param() [2/2]

```
template<typename _RealType = double>
void std::weibull_distribution< _RealType >::param (
            const param_type & __param)  [inline]
```

Sets the parameter set of the distribution.

Parameters

__param	The new parameter set of the distribution.

reset()

```
template<typename _RealType = double>
void std::weibull_distribution< _RealType >::reset ()  [inline]
```

Resets the distribution state.

6.1044.4 Friends And Related Symbol Documentation

operator==

```
template<typename _RealType = double>
bool operator== (
            const weibull_distribution< _RealType > & __d1,
            const weibull_distribution< _RealType > & __d2)  [friend]
```

Return true if two Weibull distributions have the same parameters.
The documentation for this class was generated from the following files:

- random.h
- bits/random.tcc

6.1045 std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc > Class Template Reference

```
#include <locale_conv.h>
```

Public Types

- typedef basic_string< char, char_traits< char >, _Byte_alloc > **byte_string**
- typedef wide_string::traits_type::int_type **int_type**
- typedef _Codecvt::state_type **state_type**
- typedef basic_string< _Elem, char_traits< _Elem >, _Wide_alloc > **wide_string**

Public Member Functions

- wstring_convert ()
- wstring_convert (_Codecvt *__pcvt)
- wstring_convert (_Codecvt *__pcvt, state_type __state)
- wstring_convert (const byte_string &__byte_err, const wide_string &__wide_err=wide_string())
- **wstring_convert** (const wstring_convert &)=delete
- size_t converted () const noexcept
- wstring_convert & **operator=** (const wstring_convert &)=delete
- state_type state () const

- wide_string from_bytes (char __byte)
- wide_string from_bytes (const char *__ptr)
- wide_string from_bytes (const byte_string &__str)
- wide_string from_bytes (const char *__first, const char *__last)

- byte_string to_bytes (_Elem __wchar)
- byte_string to_bytes (const _Elem *__ptr)
- byte_string to_bytes (const wide_string &__wstr)
- byte_string to_bytes (const _Elem *__first, const _Elem *__last)

6.1045.1 Detailed Description

template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>, typename _Byte_alloc = allocator<char>>
class std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >

String conversions.

6.1045.2 Constructor & Destructor Documentation

wstring_convert() [1/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert ()  [inline]
```

Default constructor.

wstring_convert() [2/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
            _Codecvt * __pcvt)  [inline], [explicit]
```

Constructor.

Parameters

__pcvt	The facet to use for conversions.

Takes ownership of __pcvt and will delete it in the destructor.

wstring_convert() [3/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
            _Codecvt * __pcvt,
            state_type __state)  [inline]
```

Construct with an initial conversion state.

Parameters

__pcvt	The facet to use for conversions.
__state	Initial conversion state.

Takes ownership of __pcvt and will delete it in the destructor. The object's conversion state will persist between conversions.

wstring_convert() [4/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::wstring_convert (
            const byte_string & __byte_err,
            const wide_string & __wide_err = wide_string())  [inline], [explicit]
```

Construct with error strings.

Parameters

__byte_err	A string to return on failed conversions.
__wide_err	A wide string to return on failed conversions.

6.1045.3 Member Function Documentation

converted()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
size_t std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::converted () const
[inline], [noexcept]
```

The number of elements successfully converted in the last conversion.

from_bytes() [1/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
            char __byte)  [inline]
```

Convert from bytes.
References from_bytes().
Referenced by from_bytes(), from_bytes(), and from_bytes().

from_bytes() [2/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
            const byte_string & __str)  [inline]
```

Convert from bytes.
References std::basic_string< _CharT, _Traits, _Alloc >::data(), from_bytes(), and std::basic_string< _CharT, _Traits, _Alloc >::size().

from_bytes() [3/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
            const char * __first,
            const char * __last)  [inline]
```

Convert from bytes.

from_bytes() [4/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
wide_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::from_bytes (
            const char * __ptr)  [inline]
```

Convert from bytes.
References from_bytes().

state()

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
state_type std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::state () const
[inline]
```

The final conversion state of the last conversion.

to_bytes() [1/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
            _Elem __wchar)  [inline]
```

Convert to bytes.
References to_bytes().
Referenced by to_bytes(), to_bytes(), and to_bytes().

to_bytes() [2/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
            const _Elem * __first,
            const _Elem * __last)  [inline]
```

Convert to bytes.

to_bytes() [3/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
            const _Elem * __ptr)  [inline]
```

Convert to bytes.
References std::basic_string< _Elem, char_traits< _Elem >, _Wide_alloc >::length(), and to_bytes().

to_bytes() [4/4]

```
template<typename _Codecvt, typename _Elem = wchar_t, typename _Wide_alloc = allocator<_Elem>,
typename _Byte_alloc = allocator<char>>
byte_string std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >::to_bytes (
            const wide_string & __wstr)  [inline]
```

Convert to bytes.
References std::basic_string< _CharT, _Traits, _Alloc >::data(), std::basic_string< _CharT, _Traits, _Alloc >::size(), and to_bytes().
The documentation for this class was generated from the following file:

- locale_conv.h

7 File Documentation

7.1 algorithm File Reference

7.1.1 Detailed Description

This is a Standard C++ Library header.

7.2 experimental/algorithm File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_sample**
- #define **_GLIBCXX_EXPERIMENTAL_ALGORITHM**

Functions

- template<typename _PopulationIterator, typename _SampleIterator, typename _Distance>
  _SampleIterator **std::experimental::sample** (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n)
- template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomNumberGenerator>
  _SampleIterator std::experimental::sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _Distance __n, _UniformRandomNumberGenerator &&__g)
- template<typename _ForwardIterator, typename _Searcher>
  _ForwardIterator **std::experimental::search** (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__searcher)
- template<typename _RandomAccessIterator>
  void **std::experimental::shuffle** (_RandomAccessIterator __first, _RandomAccessIterator __last)

7.2.1 Detailed Description

This is a TS C++ Library header.

7.3 ext/algorithm File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_ALGORITHM**

Functions

- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  std::pair< _InputIterator, _OutputIterator > **__gnu_cxx::__copy_n** (_InputIterator __first, _Size __count, _OutputIterator __result, std::input_iterator_tag)

- template<typename _RAIterator, typename _Size, typename _OutputIterator>
  std::pair< _RAIterator, _OutputIterator > **__gnu_cxx::__copy_n** (_RAIterator __first, _Size __count, _Output↩ Iterator __result, std::random_access_iterator_tag)
- template<typename _InputIterator1, typename _InputIterator2>
  int **__gnu_cxx::__lexicographical_compare_3way** (_InputIterator1 __first1, _InputIterator1 __last1, _Input↩ Iterator2 __first2, _InputIterator2 __last2)
- int **__gnu_cxx::__lexicographical_compare_3way** (const char *__first1, const char *__last1, const char *_↩ _first2, const char *__last2)
- int **__gnu_cxx::__lexicographical_compare_3way** (const unsigned char *__first1, const unsigned char *__↩ last1, const unsigned char *__first2, const unsigned char *__last2)
- template<typename _Tp>
  const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c)
- template<typename _Tp, typename _Compare>
  const _Tp & __gnu_cxx::__median (const _Tp &__a, const _Tp &__b, const _Tp &__c, _Compare __comp)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator, typename _Distance>
  _RandomAccessIterator **__gnu_cxx::__random_sample** (_InputIterator __first, _InputIterator __last, _↩ RandomAccessIterator __out, _RandomNumberGenerator &__rand, const _Distance __n)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _Distance>
  _RandomAccessIterator **__gnu_cxx::__random_sample** (_InputIterator __first, _InputIterator __last, _↩ RandomAccessIterator __out, const _Distance __n)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  std::pair< _InputIterator, _OutputIterator > __gnu_cxx::copy_n (_InputIterator __first, _Size __count, _Output↩ Iterator __result)
- template<typename _InputIterator, typename _Tp, typename _Size>
  void **__gnu_cxx::count** (_InputIterator __first, _InputIterator __last, const _Tp &__value, _Size &__n)
- template<typename _InputIterator, typename _Predicate, typename _Size>
  void **__gnu_cxx::count_if** (_InputIterator __first, _InputIterator __last, _Predicate __pred, _Size &__n)
- template<typename _RAIter>
  constexpr bool **__gnu_cxx::is_heap** (_RAIter, _RAIter)
- template<typename _FIter>
  constexpr bool **__gnu_cxx::is_sorted** (_FIter, _FIter)
- template<typename _InputIterator1, typename _InputIterator2>
  int __gnu_cxx::lexicographical_compare_3way (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _InputIterator, typename _RandomAccessIterator>
  _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random↩ AccessIterator __out_first, _RandomAccessIterator __out_last)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _RandomNumberGenerator>
  _RandomAccessIterator __gnu_cxx::random_sample (_InputIterator __first, _InputIterator __last, _Random↩ AccessIterator __out_first, _RandomAccessIterator __out_last, _RandomNumberGenerator &__rand)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance>
  _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n)
- template<typename _ForwardIterator, typename _OutputIterator, typename _Distance, typename _RandomNumberGenerator>
  _OutputIterator __gnu_cxx::random_sample_n (_ForwardIterator __first, _ForwardIterator __last, _OutputIterator __out, const _Distance __n, _RandomNumberGenerator &__rand)

7.3.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

7.4 parallel/algorithm File Reference

Macros

- #define **_PARALLEL_ALGORITHM**

7.4.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.5 any File Reference

7.5.1 Detailed Description

This is a Standard C++ Library header.

7.6 experimental/any File Reference

Classes

- class std::experimental::fundamentals_v1::any
- class std::experimental::fundamentals_v1::bad_any_cast

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_any**
- #define **_GLIBCXX_EXPERIMENTAL_ANY**

Functions

- template<typename _ValueType>
  _ValueType std::experimental::any_cast (const any &__any)
- void std::experimental::swap (any &__x, any &__y) noexcept

- template<typename _ValueType, typename enable_if< is_move_constructible< _ValueType >::value &&!is_lvalue_reference< _ValueType >::value, bool >::type = false>
  _ValueType std::experimental::any_cast (any &&__any)
- template<typename _ValueType, typename enable_if<!is_move_constructible< _ValueType >::value||is_lvalue_reference< _ValueType >::value, bool >::type = true>
  _ValueType std::experimental::any_cast (any &&__any)
- template<typename _ValueType>
  _ValueType std::experimental::any_cast (any &__any)

- template<typename _ValueType>
  _ValueType * std::experimental::any_cast (any *__any) noexcept
- template<typename _ValueType>
  const _ValueType * std::experimental::any_cast (const any *__any) noexcept

7.6.1 Detailed Description

This is a TS C++ Library header.

7.7 array File Reference

Classes

- struct std::array< _Tp, _Nm >
- struct std::tuple_element< _Ind, array< _Tp, _Nm > >
- struct std::tuple_size< array< _Tp, _Nm > >

Namespaces

- namespace std

Functions

- template<typename _Tp, typename... _Up>
  std::array (_Tp, _Up...) -> array< enable_if_t<(is_same_v< _Tp, _Up > &&...), _Tp >, 1+sizeof...(_Up)>
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr _Tp && **std::get** (array< _Tp, _Nm > &&__arr) noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr _Tp & **std::get** (array< _Tp, _Nm > &__arr) noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr const _Tp && **std::get** (const array< _Tp, _Nm > &&__arr) noexcept
- template<std::size_t _Int, typename _Tp, std::size_t _Nm>
  constexpr const _Tp & **std::get** (const array< _Tp, _Nm > &__arr) noexcept
- template<typename _Tp, size_t _Nm>
  constexpr __detail::__synth3way_t< _Tp > **std::operator<=>** (const array< _Tp, _Nm > &__a, const array< _Tp, _Nm > &__b)
- template<typename _Tp, std::size_t _Nm>
  constexpr bool **std::operator==** (const array< _Tp, _Nm > &__one, const array< _Tp, _Nm > &__two)
- template<typename _Tp, std::size_t _Nm>
  __enable_if_t<!__array_traits< _Tp, _Nm >::_Is_swappable::value > **std::swap** (array< _Tp, _Nm > &, array< _Tp, _Nm > &)=delete
- template<typename _Tp, std::size_t _Nm>
  constexpr __enable_if_t< __array_traits< _Tp, _Nm >::_Is_swappable::value > **std::swap** (array< _Tp, _Nm > &__one, array< _Tp, _Nm > &__two) noexcept(noexcept(__one.swap(__two)))

Variables

- template<typename _Tp, size_t _Nm>
  constexpr size_t **std::tuple_size_v**< **array**< **_Tp, _Nm** > >
- template<typename _Tp, size_t _Nm>
  constexpr size_t **std::tuple_size_v**< **const array**< **_Tp, _Nm** > >

7.7.1 Detailed Description

This is a Standard C++ Library header.

7.8 experimental/array File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_make_array**
- #define **_GLIBCXX_EXPERIMENTAL_ARRAY**

Functions

- template<typename _Tp, size_t _Nm, size_t... _Idx>
  constexpr array< remove_cv_t< _Tp >, _Nm > **std::experimental::__to_array** (_Tp(&__a)[_Nm]↩
  , index_sequence< _Idx... >)
- template<typename _Dest = void, typename... _Types>
  constexpr array< typename __make_array_elem< _Dest, _Types... >::type, sizeof...(_Types)> std::experimental::make_array (_Types &&... __t)
- template<typename _Tp, size_t _Nm>
  constexpr array< remove_cv_t< _Tp >, _Nm > std::experimental::to_array (_Tp(&__a)[_Nm]) noexcept(is_nothrow_constructible< remove_cv_t< _Tp >, _Tp & >::value)

7.8.1 Detailed Description

This is a TS C++ Library header.

7.9 atomic File Reference

Classes

- struct std::atomic< _Tp >
- struct std::atomic< _Tp * >
- struct std::atomic< bool >
- struct std::atomic< char >
- struct std::atomic< char16_t >
- struct std::atomic< char32_t >
- struct std::atomic< int >
- struct std::atomic< long >
- struct std::atomic< long long >
- struct std::atomic< short >
- struct std::atomic< signed char >
- struct std::atomic< unsigned char >
- struct std::atomic< unsigned int >
- struct std::atomic< unsigned long >
- struct std::atomic< unsigned long long >
- struct std::atomic< unsigned short >
- struct std::atomic< wchar_t >

Namespaces

- namespace std

Typedefs

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

Functions

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

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

- template<typename _ITp>
  _ITp **std::atomic_load_explicit** (const atomic< _ITp > *__a, memory_order __m) noexcept
- template<typename _ITp>
  _ITp **std::atomic_load_explicit** (const volatile atomic< _ITp > *__a, memory_order __m) noexcept
- template<typename _ITp>
  void **std::atomic_store** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **std::atomic_store** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i) noexcept
- template<typename _ITp>
  void **std::atomic_store_explicit** (atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept
- template<typename _ITp>
  void **std::atomic_store_explicit** (volatile atomic< _ITp > *__a, __atomic_val_t< _ITp > __i, memory_order __m) noexcept

7.9.1 Detailed Description

This is a Standard C++ Library header.

7.10 auto_ptr.h File Reference

Classes

- class std::auto_ptr< _Tp >
- struct std::auto_ptr_ref< _Tp1 >

Namespaces

- namespace std

7.10.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.11 backward_warning.h File Reference

7.11.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

7.12 hash_fun.h File Reference

Namespaces

- namespace __gnu_cxx

Typedefs

- typedef __SIZE_TYPE__ **__gnu_cxx::size_t**

Functions

- size_t **__gnu_cxx::__stl_hash_string** (const char *__s)

7.12.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

7.13 hash_map File Reference

Classes

- class __gnu_cxx::allocator< _Tp >
- struct __gnu_cxx::equal_to< _Tp >
- class __gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >
- struct __gnu_cxx::pair< _T1, _T2 >

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_BACKWARD_HASH_MAP**

Functions

- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  bool **__gnu_cxx::operator!=** (const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>
  bool **__gnu_cxx::operator!=** (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  bool **__gnu_cxx::operator==** (const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, const hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HF, class _EqKey, class _Alloc>
  bool **__gnu_cxx::operator==** (const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm1, const hash_multimap< _Key, _Tp, _HF, _EqKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  void **__gnu_cxx::swap** (hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, hash_map< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)
- template<class _Key, class _Tp, class _HashFn, class _EqlKey, class _Alloc>
  void **__gnu_cxx::swap** (hash_multimap< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm1, hash_multimap< _Key, _Tp, _HashFn, _EqlKey, _Alloc > &__hm2)

7.13.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

7.14 hash_set File Reference

Classes

- class __gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >
- class __gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_BACKWARD_HASH_SET**

Functions

- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  bool **__gnu_cxx::operator!=** (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>
  bool **__gnu_cxx::operator!=** (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  bool **__gnu_cxx::operator==** (const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Value, class _HashFcn, class _EqualKey, class _Alloc>
  bool **__gnu_cxx::operator==** (const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs1, const hash_set< _Value, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  void **__gnu_cxx::swap** (hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_multiset< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)
- template<class _Val, class _HashFcn, class _EqualKey, class _Alloc>
  void **__gnu_cxx::swap** (hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs1, hash_set< _Val, _HashFcn, _EqualKey, _Alloc > &__hs2)

7.14.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

7.15 strstream File Reference

Namespaces

- namespace std

Macros

- #define **_BACKWARD_STRSTREAM**
- #define **_GLIBCXX_STRSTREAM_DEPR**(A, B)

7.15.1 Detailed Description

This is a Standard C++ Library header.

7.16 barrier File Reference

7.16.1 Detailed Description

This is a Standard C++ Library header.

7.17 bit File Reference

Namespaces

- namespace std

7.17.1 Detailed Description

This is a Standard C++ Library header.

7.18 bits/algorithmfwd.h File Reference

Namespaces

- namespace std

Functions

- template<typename _FIter>
  constexpr _FIter **std::adjacent_find** (_FIter, _FIter)
- template<typename _FIter, typename _BinaryPredicate>
  constexpr _FIter **std::adjacent_find** (_FIter, _FIter, _BinaryPredicate)
- template<typename _IIter, typename _Predicate>
  constexpr bool **std::all_of** (_IIter, _IIter, _Predicate)
- template<typename _IIter, typename _Predicate>
  constexpr bool **std::any_of** (_IIter, _IIter, _Predicate)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr bool **std::binary_search** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type, typename _Compare>
  constexpr bool **std::binary_search** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _Tp>
  constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)
- template<typename _IIter, typename _OIter>
  constexpr _OIter **std::copy** (_IIter, _IIter, _OIter)
- template<typename _BIter1, typename _BIter2>
  constexpr _BIter2 **std::copy_backward** (_BIter1, _BIter1, _BIter2)
- template<typename _IIter, typename _OIter, typename _Predicate>
  constexpr _OIter **std::copy_if** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _IIter, typename _Size, typename _OIter>
  constexpr _OIter **std::copy_n** (_IIter, _Size, _OIter)
- template<typename _IIter, typename _Tp = typename iterator_traits<_IIter>::value_type>
  constexpr iterator_traits< _IIter >::difference_type **std::count** (_IIter, _IIter, const _Tp &)
- template<typename _IIter, typename _Predicate>
  constexpr iterator_traits< _IIter >::difference_type **std::count_if** (_IIter, _IIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::equal** (_IIter1, _IIter1, _IIter2)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr pair< _FIter, _FIter > **std::equal_range** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type, typename _Compare>
  constexpr pair< _FIter, _FIter > **std::equal_range** (_FIter, _FIter, const _Tp &, _Compare)

- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr void **std::fill** (_FIter, _FIter, const _Tp &)
- template<typename _OIter, typename _Size, typename _Tp = typename iterator_traits<_OIter>::value_type>
  constexpr _OIter **std::fill_n** (_OIter, _Size, const _Tp &)
- template<typename _IIter, typename _Tp = typename iterator_traits<_IIter>::value_type>
  constexpr _IIter **std::find** (_IIter, _IIter, const _Tp &)
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter1 **std::find_end** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr _FIter1 **std::find_end** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter1 **std::find_first_of** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr _FIter1 **std::find_first_of** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _IIter, typename _Predicate>
  constexpr _IIter **std::find_if** (_IIter, _IIter, _Predicate)
- template<typename _IIter, typename _Predicate>
  constexpr _IIter **std::find_if_not** (_IIter, _IIter, _Predicate)
- template<typename _IIter, typename _Funct>
  constexpr _Funct **std::for_each** (_IIter, _IIter, _Funct)
- template<typename _FIter, typename _Generator>
  constexpr void **std::generate** (_FIter, _FIter, _Generator)
- template<typename _OIter, typename _Size, typename _Generator>
  constexpr _OIter **std::generate_n** (_OIter, _Size, _Generator)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::includes** (_IIter1, _IIter1, _IIter2, _IIter2)
- template<typename _IIter1, typename _IIter2, typename _Compare>
  constexpr bool **std::includes** (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)
- template<typename _BIter>
  constexpr void **std::inplace_merge** (_BIter, _BIter, _BIter)
- template<typename _BIter, typename _Compare>
  constexpr void **std::inplace_merge** (_BIter, _BIter, _BIter, _Compare)
- template<typename _RAIter>
  constexpr bool **std::is_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr bool **std::is_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RAIter>
  constexpr _RAIter **std::is_heap_until** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr _RAIter **std::is_heap_until** (_RAIter, _RAIter, _Compare)
- template<typename _IIter, typename _Predicate>
  constexpr bool **std::is_partitioned** (_IIter, _IIter, _Predicate)
- template<typename _FIter1, typename _FIter2>
  constexpr bool **std::is_permutation** (_FIter1, _FIter1, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr bool **std::is_permutation** (_FIter1, _FIter1, _FIter2, _BinaryPredicate)
- template<typename _FIter>
  constexpr bool **std::is_sorted** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr bool **std::is_sorted** (_FIter, _FIter, _Compare)
- template<typename _FIter>
  constexpr _FIter **std::is_sorted_until** (_FIter, _FIter)

- template<typename _FIter, typename _Compare>
  constexpr _FIter **std::is_sorted_until** (_FIter, _FIter, _Compare)
- template<typename _FIter1, typename _FIter2>
  constexpr void **std::iter_swap** (_FIter1, _FIter2)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::lexicographical_compare** (_IIter1, _IIter1, _IIter2, _IIter2)
- template<typename _IIter1, typename _IIter2, typename _Compare>
  constexpr bool **std::lexicographical_compare** (_IIter1, _IIter1, _IIter2, _IIter2, _Compare)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr _FIter **std::lower_bound** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type, typename _Compare>
  constexpr _FIter **std::lower_bound** (_FIter, _FIter, const _Tp &, _Compare)
- template<typename _RAIter>
  constexpr void **std::make_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::make_heap** (_RAIter, _RAIter, _Compare)
- template<typename _Tp>
  constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp>
  constexpr _Tp **std::max** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr _Tp **std::max** (initializer_list< _Tp >, _Compare)
- template<typename _FIter>
  constexpr _FIter **std::max_element** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr _FIter **std::max_element** (_FIter, _FIter, _Compare)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **std::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **std::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _Tp>
  constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp>
  constexpr _Tp **std::min** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr _Tp **std::min** (initializer_list< _Tp >, _Compare)
- template<typename _FIter>
  constexpr _FIter **std::min_element** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr _FIter **std::min_element** (_FIter, _FIter, _Compare)
- template<typename _Tp>
  constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp>
  constexpr pair< _Tp, _Tp > **std::minmax** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr pair< _Tp, _Tp > **std::minmax** (initializer_list< _Tp >, _Compare)

- template<typename _FIter>
  constexpr pair< _FIter, _FIter > **std::minmax_element** (_FIter, _FIter)
- template<typename _FIter, typename _Compare>
  constexpr pair< _FIter, _FIter > **std::minmax_element** (_FIter, _FIter, _Compare)
- template<typename _IIter1, typename _IIter2>
  constexpr pair< _IIter1, _IIter2 > **std::mismatch** (_IIter1, _IIter1, _IIter2)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr pair< _IIter1, _IIter2 > **std::mismatch** (_IIter1, _IIter1, _IIter2, _BinaryPredicate)
- template<typename _BIter>
  constexpr bool **std::next_permutation** (_BIter, _BIter)
- template<typename _BIter, typename _Compare>
  constexpr bool **std::next_permutation** (_BIter, _BIter, _Compare)
- template<typename _IIter, typename _Predicate>
  constexpr bool **std::none_of** (_IIter, _IIter, _Predicate)
- template<typename _RAIter>
  constexpr void **std::nth_element** (_RAIter, _RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::nth_element** (_RAIter, _RAIter, _RAIter, _Compare)
- template<typename _RAIter>
  constexpr void **std::partial_sort** (_RAIter, _RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::partial_sort** (_RAIter, _RAIter, _RAIter, _Compare)
- template<typename _IIter, typename _RAIter>
  constexpr _RAIter **std::partial_sort_copy** (_IIter, _IIter, _RAIter, _RAIter)
- template<typename _IIter, typename _RAIter, typename _Compare>
  constexpr _RAIter **std::partial_sort_copy** (_IIter, _IIter, _RAIter, _RAIter, _Compare)
- template<typename _BIter, typename _Predicate>
  constexpr _BIter **std::partition** (_BIter, _BIter, _Predicate)
- template<typename _IIter, typename _OIter1, typename _OIter2, typename _Predicate>
  constexpr pair< _OIter1, _OIter2 > **std::partition_copy** (_IIter, _IIter, _OIter1, _OIter2, _Predicate)
- template<typename _FIter, typename _Predicate>
  constexpr _FIter **std::partition_point** (_FIter, _FIter, _Predicate)
- template<typename _RAIter>
  constexpr void **std::pop_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::pop_heap** (_RAIter, _RAIter, _Compare)
- template<typename _BIter>
  constexpr bool **std::prev_permutation** (_BIter, _BIter)
- template<typename _BIter, typename _Compare>
  constexpr bool **std::prev_permutation** (_BIter, _BIter, _Compare)
- template<typename _RAIter>
  constexpr void **std::push_heap** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::push_heap** (_RAIter, _RAIter, _Compare)
- template<typename _RAIter>
  void **std::random_shuffle** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Generator>
  void **std::random_shuffle** (_RAIter, _RAIter, _Generator &&)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr _FIter **std::remove** (_FIter, _FIter, const _Tp &)
- template<typename _IIter, typename _OIter, typename _Tp = typename iterator_traits<_IIter>::value_type>
  constexpr _OIter **std::remove_copy** (_IIter, _IIter, _OIter, const _Tp &)

- template<typename _IIter, typename _OIter, typename _Predicate>
  constexpr _OIter **std::remove_copy_if** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _FIter, typename _Predicate>
  constexpr _FIter **std::remove_if** (_FIter, _FIter, _Predicate)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr void **std::replace** (_FIter, _FIter, const _Tp &, const _Tp &)
- template<typename _IIter, typename _OIter, typename _Tp>
  constexpr _OIter **std::replace_copy** (_IIter, _IIter, _OIter, const _Tp &, const _Tp &)
- template<typename _Iter, typename _OIter, typename _Predicate, typename _Tp = typename iterator_traits<_OIter>::value_type>
  constexpr _OIter **std::replace_copy_if** (_Iter, _Iter, _OIter, _Predicate, const _Tp &)
- template<typename _FIter, typename _Predicate, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr void **std::replace_if** (_FIter, _FIter, _Predicate, const _Tp &)
- template<typename _BIter>
  constexpr void **std::reverse** (_BIter, _BIter)
- template<typename _BIter, typename _OIter>
  constexpr _OIter **std::reverse_copy** (_BIter, _BIter, _OIter)
- template<typename _FIter>
  constexpr _FIter **std::rotate** (_FIter, _FIter, _FIter)
- template<typename _FIter, typename _OIter>
  constexpr _OIter **std::rotate_copy** (_FIter, _FIter, _FIter, _OIter)
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter1 **std::search** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2, typename _BinaryPredicate>
  constexpr _FIter1 **std::search** (_FIter1, _FIter1, _FIter2, _FIter2, _BinaryPredicate)
- template<typename _FIter, typename _Size, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr _FIter **std::search_n** (_FIter, _FIter, _Size, const _Tp &)
- template<typename _FIter, typename _Size, typename _Tp = typename iterator_traits<_FIter>::value_type, typename _BinaryPredicate>
  constexpr _FIter **std::search_n** (_FIter, _FIter, _Size, const _Tp &, _BinaryPredicate)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **std::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **std::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **std::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **std::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **std::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **std::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  constexpr _OIter **std::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  constexpr _OIter **std::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _RAIter, typename _UGenerator>
  void **std::shuffle** (_RAIter, _RAIter, _UGenerator &&)
- template<typename _RAIter>
  constexpr void **std::sort** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::sort** (_RAIter, _RAIter, _Compare)
- template<typename _RAIter>
  constexpr void **std::sort_heap** (_RAIter, _RAIter)

- template<typename _RAIter, typename _Compare>
  constexpr void **std::sort_heap** (_RAIter, _RAIter, _Compare)
- template<typename _BIter, typename _Predicate>
  constexpr _BIter **std::stable_partition** (_BIter, _BIter, _Predicate)
- template<typename _RAIter>
  constexpr void **std::stable_sort** (_RAIter, _RAIter)
- template<typename _RAIter, typename _Compare>
  constexpr void **std::stable_sort** (_RAIter, _RAIter, _Compare)
- template<typename _FIter1, typename _FIter2>
  constexpr _FIter2 **std::swap_ranges** (_FIter1, _FIter1, _FIter2)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  constexpr _OIter **std::transform** (_IIter, _IIter, _OIter, _UnaryOperation)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BinaryOperation>
  constexpr _OIter **std::transform** (_IIter1, _IIter1, _IIter2, _OIter, _BinaryOperation)
- template<typename _FIter>
  constexpr _FIter **std::unique** (_FIter, _FIter)
- template<typename _FIter, typename _BinaryPredicate>
  constexpr _FIter **std::unique** (_FIter, _FIter, _BinaryPredicate)
- template<typename _IIter, typename _OIter>
  constexpr _OIter **std::unique_copy** (_IIter, _IIter, _OIter)
- template<typename _IIter, typename _OIter, typename _BinaryPredicate>
  constexpr _OIter **std::unique_copy** (_IIter, _IIter, _OIter, _BinaryPredicate)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type>
  constexpr _FIter **std::upper_bound** (_FIter, _FIter, const _Tp &)
- template<typename _FIter, typename _Tp = typename iterator_traits<_FIter>::value_type, typename _Compare>
  constexpr _FIter **std::upper_bound** (_FIter, _FIter, const _Tp &, _Compare)

7.18.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <algorithm>.

7.19 parallel/algorithmfwd.h File Reference

Namespaces

- namespace std
- namespace std::__parallel

Functions

- template<typename _FIter, typename _BiPredicate, typename _IterTag>
  _FIter **std::__parallel::__adjacent_find_switch** (_FIter, _FIter, _BiPredicate, _IterTag)
- template<typename _FIter, typename _IterTag>
  _FIter **std::__parallel::__adjacent_find_switch** (_FIter, _FIter, _IterTag)
- template<typename _RAIter>
  _RAIter **std::__parallel::__adjacent_find_switch** (_RAIter __begin, _RAIter __end, random_access_iterator_tag)
- template<typename _RAIter, typename _BiPredicate>
  _RAIter **std::__parallel::__adjacent_find_switch** (_RAIter, _RAIter, _BiPredicate, random_access_iterator_tag)
- template<typename _IIter, typename _Predicate, typename _IterTag>
  iterator_traits< _IIter >::difference_type **std::__parallel::__count_if_switch** (_IIter, _IIter, _Predicate, _IterTag)
- template<typename _RAIter, typename _Predicate>
  iterator_traits< _RAIter >::difference_type **std::__parallel::__count_if_switch** (_RAIter __begin, _RAIter __↩
  end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)

- template<typename _IIter, typename _Tp, typename _IterTag>
  iterator_traits< _IIter >::difference_type **std::__parallel::__count_switch** (_IIter, _IIter, const _Tp &, _IterTag)
- template<typename _RAIter, typename _Tp>
  iterator_traits< _RAIter >::difference_type **std::__parallel::__count_switch** (_RAIter __begin, _RAIter __end, const _Tp &__value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _FIter, typename _BiPredicate, typename _IterTag1, typename _IterTag2>
  _IIter **std::__parallel::__find_first_of_switch** (_IIter, _IIter, _FIter, _FIter, _BiPredicate, _IterTag1, _IterTag2)
- template<typename _IIter, typename _FIter, typename _IterTag1, typename _IterTag2>
  _IIter **std::__parallel::__find_first_of_switch** (_IIter, _IIter, _FIter, _FIter, _IterTag1, _IterTag2)
- template<typename _RAIter, typename _FIter, typename _BiPredicate, typename _IterTag>
  _RAIter **std::__parallel::__find_first_of_switch** (_RAIter, _RAIter, _FIter, _FIter, _BiPredicate, random_access_iterator_tag, _IterTag)
- template<typename _IIter, typename _Predicate, typename _IterTag>
  _IIter **std::__parallel::__find_if_switch** (_IIter, _IIter, _Predicate, _IterTag)
- template<typename _RAIter, typename _Predicate>
  _RAIter **std::__parallel::__find_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _IIter, typename _Tp, typename _IterTag>
  _IIter **std::__parallel::__find_switch** (_IIter, _IIter, const _Tp &, _IterTag)
- template<typename _RAIter, typename _Tp>
  _RAIter **std::__parallel::__find_switch** (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)
- template<typename _IIter, typename _Function, typename _IterTag>
  _Function **std::__parallel::__for_each_switch** (_IIter, _IIter, _Function, _IterTag)
- template<typename _RAIter, typename _Function>
  _Function **std::__parallel::__for_each_switch** (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _OIter, typename _Size, typename _Generator, typename _IterTag>
  _OIter **std::__parallel::__generate_n_switch** (_OIter, _Size, _Generator, _IterTag)
- template<typename _RAIter, typename _Size, typename _Generator>
  _RAIter **std::__parallel::__generate_n_switch** (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Generator, typename _IterTag>
  void **std::__parallel::__generate_switch** (_FIter, _FIter, _Generator, _IterTag)
- template<typename _RAIter, typename _Generator>
  void **std::__parallel::__generate_switch** (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, typename _IterTag2>
  bool **std::__parallel::__lexicographical_compare_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _Predicate, _Iter↩
  Tag1, _IterTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  bool **std::__parallel::__lexicographical_compare_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 _↩
  _begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter, typename _Compare, typename _IterTag>
  _FIter **std::__parallel::__max_element_switch** (_FIter, _FIter, _Compare, _IterTag)
- template<typename _RAIter, typename _Compare>
  _RAIter **std::__parallel::__max_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **std::__parallel::__merge_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, _IterTag1, _IterTag2, _IterTag3)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  _OIter **std::__parallel::__merge_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)

- template<typename _FIter, typename _Compare, typename _IterTag>
  _FIter **std::__parallel::__min_element_switch** (_FIter, _FIter, _Compare, _IterTag)
- template<typename _RAIter, typename _Compare>
  _RAIter **std::__parallel::__min_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IterTag1, typename _IterTag2>
  pair< _IIter1, _IIter2 > **std::__parallel::__mismatch_switch** (_IIter1, _IIter1, _IIter2, _Predicate, _IterTag1, _IterTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  pair< _RAIter1, _RAIter2 > **std::__parallel::__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter, typename _Predicate, typename _IterTag>
  _FIter **std::__parallel::__partition_switch** (_FIter, _FIter, _Predicate, _IterTag)
- template<typename _RAIter, typename _Predicate>
  _RAIter **std::__parallel::__partition_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _FIter, typename _Predicate, typename _Tp, typename _IterTag>
  void **std::__parallel::__replace_if_switch** (_FIter, _FIter, _Predicate, const _Tp &, _IterTag)
- template<typename _RAIter, typename _Predicate, typename _Tp>
  void **std::__parallel::__replace_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Tp, typename _IterTag>
  void **std::__parallel::__replace_switch** (_FIter, _FIter, const _Tp &, const _Tp &, _IterTag)
- template<typename _RAIter, typename _Tp>
  void **std::__parallel::__replace_switch** (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate, typename _IterTag>
  _FIter **std::__parallel::__search_n_switch** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, _IterTag)
- template<typename _RAIter, typename _Integer, typename _Tp, typename _BiPredicate>
  _RAIter **std::__parallel::__search_n_switch** (_RAIter, _RAIter, _Integer, const _Tp &, _BiPredicate, random_access_iterator_tag)
- template<typename _FIter1, typename _FIter2, typename _BiPredicate, typename _IterTag1, typename _IterTag2>
  _FIter1 **std::__parallel::__search_switch** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, _IterTag1, _IterTag2)
- template<typename _FIter1, typename _FIter2, typename _IterTag1, typename _IterTag2>
  _FIter1 **std::__parallel::__search_switch** (_FIter1, _FIter1, _FIter2, _FIter2, _IterTag1, _IterTag2)
- template<typename _RAIter1, typename _RAIter2>
  _RAIter1 **std::__parallel::__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter1, typename _RAIter2, typename _BiPredicate>
  _RAIter1 **std::__parallel::__search_switch** (_RAIter1, _RAIter1, _RAIter2, _RAIter2, _BiPredicate, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **std::__parallel::__set_difference_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **std::__parallel::__set_intersection_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)

- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_intersection_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **std::__parallel::__set_symmetric_difference_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_symmetric_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OIter, typename _IterTag1, typename _IterTag2, typename _IterTag3>
  _OIter **std::__parallel::__set_union_switch** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, _IterTag1, _IterTag2, _IterTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_union_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _OIter, typename _UnaryOperation, typename _IterTag1, typename _IterTag2>
  _OIter **std::__parallel::__transform1_switch** (_IIter, _IIter, _OIter, _UnaryOperation, _IterTag1, _IterTag2)
- template<typename _RAIIter, typename _RAOIter, typename _UnaryOperation>
  _RAOIter **std::__parallel::__transform1_switch** (_RAIIter, _RAIIter, _RAOIter, _UnaryOperation, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_balanced)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation, typename _Tag1, typename _Tag2, typename _Tag3>
  _OIter **std::__parallel::__transform2_switch** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, _Tag1, _Tag2, _Tag3)
- template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BiOperation>
  _RAIter3 **std::__parallel::__transform2_switch** (_RAIter1, _RAIter1, _RAIter2, _RAIter3, _BiOperation, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_balanced)
- template<typename _IIter, typename _OIter, typename _Predicate, typename _IterTag1, typename _IterTag2>
  _OIter **std::__parallel::__unique_copy_switch** (_IIter, _IIter, _OIter, _Predicate, _IterTag1, _IterTag2)
- template<typename _RAIter, typename _RandomAccess_OIter, typename _Predicate>
  _RandomAccess_OIter **std::__parallel::__unique_copy_switch** (_RAIter, _RAIter, _RandomAccess_OIter, Predicate, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIter>
  _FIter **std::__parallel::adjacent_find** (_FIter, _FIter)
- template<typename _FIter>
  _FIter **std::__parallel::adjacent_find** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _BiPredicate>
  _FIter **std::__parallel::adjacent_find** (_FIter, _FIter, _BiPredicate)
- template<typename _FIter, typename _BiPredicate>
  _FIter **std::__parallel::adjacent_find** (_FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__value)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::_Parallelism __parallelism_tag)

- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__value, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1, typename _IIter2>
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  constexpr bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp>
  _IIter **std::__parallel::find** (_IIter __begin, _IIter __end, const _Tp &__val)
- template<typename _IIter, typename _Tp>
  _IIter **std::__parallel::find** (_IIter __begin, _IIter __end, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIter>
  _IIter **std::__parallel::find_first_of** (_IIter, _IIter, _FIter, _FIter)
- template<typename _IIter, typename _FIter>
  _IIter **std::__parallel::find_first_of** (_IIter, _IIter, _FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIter, typename _BiPredicate>
  _IIter **std::__parallel::find_first_of** (_IIter, _IIter, _FIter, _FIter, _BiPredicate)
- template<typename _IIter, typename _FIter, typename _BiPredicate>
  _IIter **std::__parallel::find_first_of** (_IIter, _IIter, _FIter, _FIter, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Predicate>
  _IIter **std::__parallel::find_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  _IIter **std::__parallel::find_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Function>
  _Function **std::__parallel::for_each** (_IIter __begin, _IIter __end, _Function __f, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Function>
  _Function **std::__parallel::for_each** (_IIter, _IIter, _Function)
- template<typename _Iterator, typename _Function>
  _Function **std::__parallel::for_each** (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter, typename _Generator>
  void **std::__parallel::generate** (_FIter, _FIter, _Generator)
- template<typename _FIter, typename _Generator>
  void **std::__parallel::generate** (_FIter, _FIter, _Generator, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Generator>
  void **std::__parallel::generate** (_FIter, _FIter, _Generator, __gnu_parallel::sequential_tag)
- template<typename _OIter, typename _Size, typename _Generator>
  _OIter **std::__parallel::generate_n** (_OIter, _Size, _Generator)

- template<typename _OIter, typename _Size, typename _Generator>
  _OIter **std::__parallel::generate_n** (_OIter, _Size, _Generator, __gnu_parallel::_Parallelism)
- template<typename _OIter, typename _Size, typename _Generator>
  _OIter **std::__parallel::generate_n** (_OIter, _Size, _Generator, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template<typename _IIter1, typename _IIter2>
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 _↩
  _end2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  constexpr bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 _↩
  _end2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _FIter>
  _FIter **std::__parallel::max_element** (_FIter, _FIter)
- template<typename _FIter>
  _FIter **std::__parallel::max_element** (_FIter, _FIter, __gnu_parallel::_Parallelism)
- template<typename _FIter>
  _FIter **std::__parallel::max_element** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Compare>
  _FIter **std::__parallel::max_element** (_FIter, _FIter, _Compare)
- template<typename _FIter, typename _Compare>
  _FIter **std::__parallel::max_element** (_FIter, _FIter, _Compare, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Compare>
  _FIter **std::__parallel::max_element** (_FIter, _FIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  _OIter **std::__parallel::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Compare>
  _OIter **std::__parallel::merge** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _FIter>
  _FIter **std::__parallel::min_element** (_FIter, _FIter)
- template<typename _FIter>
  _FIter **std::__parallel::min_element** (_FIter, _FIter, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIter>
  _FIter **std::__parallel::min_element** (_FIter, _FIter, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Compare>
  _FIter **std::__parallel::min_element** (_FIter, _FIter, _Compare)
- template<typename _FIter, typename _Compare>
  _FIter **std::__parallel::min_element** (_FIter, _FIter, _Compare, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Compare>
  _FIter **std::__parallel::min_element** (_FIter, _FIter, _Compare, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)

- template<typename _IIter1, typename _IIter2, typename _Predicate>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _↩
  Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _↩
  Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Predicate>
  _FIter **std::__parallel::partition** (_FIter, _FIter, _Predicate)
- template<typename _FIter, typename _Predicate>
  _FIter **std::__parallel::partition** (_FIter, _FIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Tp>
  void **std::__parallel::replace** (_FIter, _FIter, const _Tp &, const _Tp &)
- template<typename _FIter, typename _Tp>
  void **std::__parallel::replace** (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Tp>
  void **std::__parallel::replace** (_FIter, _FIter, const _Tp &, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Predicate, typename _Tp>
  void **std::__parallel::replace_if** (_FIter, _FIter, _Predicate, const _Tp &)
- template<typename _FIter, typename _Predicate, typename _Tp>
  void **std::__parallel::replace_if** (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::_Parallelism)
- template<typename _FIter, typename _Predicate, typename _Tp>
  void **std::__parallel::replace_if** (_FIter, _FIter, _Predicate, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIter1, typename _FIter2>
  _FIter1 **std::__parallel::search** (_FIter1, _FIter1, _FIter2, _FIter2)
- template<typename _FIter1, typename _FIter2>
  _FIter1 **std::__parallel::search** (_FIter1, _FIter1, _FIter2, _FIter2, __gnu_parallel::sequential_tag)

- template<typename _FIter1, typename _FIter2, typename _BiPredicate>
  constexpr _FIter1 **std::__parallel::search** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate)
- template<typename _FIter1, typename _FIter2, typename _BiPredicate>
  _FIter1 **std::__parallel::search** (_FIter1, _FIter1, _FIter2, _FIter2, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Integer, typename _Tp>
  _FIter **std::__parallel::search_n** (_FIter, _FIter, _Integer, const _Tp &)
- template<typename _FIter, typename _Integer, typename _Tp>
  _FIter **std::__parallel::search_n** (_FIter, _FIter, _Integer, const _Tp &, __gnu_parallel::sequential_tag)
- template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate>
  _FIter **std::__parallel::search_n** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate)
- template<typename _FIter, typename _Integer, typename _Tp, typename _BiPredicate>
  _FIter **std::__parallel::search_n** (_FIter, _FIter, _Integer, const _Tp &, _BiPredicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_intersection** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_symmetric_difference** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter)
- template<typename _IIter1, typename _IIter2, typename _OIter>
  _OIter **std::__parallel::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::set_union** (_IIter1, _IIter1, _IIter2, _IIter2, _OIter, _Predicate, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)

- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  _OIter **std::__parallel::transform** (_IIter, _IIter, _OIter, _UnaryOperation)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  _OIter **std::__parallel::transform** (_IIter, _IIter, _OIter, _UnaryOperation, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _OIter, typename _UnaryOperation>
  _OIter **std::__parallel::transform** (_IIter, _IIter, _OIter, _UnaryOperation, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation>
  _OIter **std::__parallel::transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation>
  _OIter **std::__parallel::transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _OIter, typename _BiOperation>
  _OIter **std::__parallel::transform** (_IIter1, _IIter1, _IIter2, _OIter, _BiOperation, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::unique_copy** (_IIter, _IIter, _OIter)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::unique_copy** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::unique_copy** (_IIter, _IIter, _OIter, _Predicate)
- template<typename _IIter, typename _OIter, typename _Predicate>
  _OIter **std::__parallel::unique_copy** (_IIter, _IIter, _OIter, _Predicate, __gnu_parallel::sequential_tag)

7.19.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

7.20 align.h File Reference

Namespaces

- namespace std

Functions

- void * std::align (size_t __align, size_t __size, void *&__ptr, size_t &__space) noexcept
- template<size_t _Align, class _Tp>
  constexpr _Tp * std::assume_aligned (_Tp *__ptr) noexcept
- template<size_t _Align, class _Tp>
  bool std::is_sufficiently_aligned (_Tp *__ptr)

7.20.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.21 bits/alloc_traits.h File Reference

Classes

- struct std::allocator_traits< _Alloc >
- struct std::allocator_traits< allocator< _Tp > >
- struct std::allocator_traits< allocator< void > >

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_NO_ALLOC_TRAITS_SPECIALIZATIONS**

7.21.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.22 ext/alloc_traits.h File Reference

Classes

- struct __gnu_cxx::__alloc_traits< _Alloc, typename >

Namespaces

- namespace __gnu_cxx

7.22.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.23 allocated_ptr.h File Reference

Namespaces

- namespace std

7.23.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.24 allocator.h File Reference

Classes

- class std::allocator< _Tp >
- class std::allocator< void >

Namespaces

- namespace std

7.24.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.25 atomic_base.h File Reference

Classes

- struct std::atomic_flag

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_ALWAYS_INLINE**
- #define **ATOMIC_FLAG_INIT**
- #define **ATOMIC_VAR_INIT**(_VI)

Typedefs

- typedef unsigned char **std::__atomic_flag_data_type**

Enumerations

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

Functions

- void **std::atomic_signal_fence** (memory_order __m) noexcept
- void **std::atomic_thread_fence** (memory_order __m) noexcept
- template<typename _Tp>
  _Tp std::kill_dependency (_Tp __y) noexcept
- constexpr memory_order **std::operator&** (memory_order __m, __memory_order_modifier __mod) noexcept
- constexpr memory_order **std::operator|** (memory_order __m, __memory_order_modifier __mod) noexcept

Variables

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

7.25.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <atomic>.

7.26 atomic_futex.h File Reference

Namespaces

- namespace std

7.26.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

7.27 atomic_lockfree_defines.h File Reference

Macros

- #define ATOMIC_BOOL_LOCK_FREE
- #define **ATOMIC_CHAR16_T_LOCK_FREE**
- #define **ATOMIC_CHAR32_T_LOCK_FREE**
- #define **ATOMIC_CHAR_LOCK_FREE**
- #define **ATOMIC_INT_LOCK_FREE**
- #define **ATOMIC_LLONG_LOCK_FREE**
- #define **ATOMIC_LONG_LOCK_FREE**
- #define **ATOMIC_POINTER_LOCK_FREE**
- #define **ATOMIC_SHORT_LOCK_FREE**
- #define **ATOMIC_WCHAR_T_LOCK_FREE**

7.27.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <atomic>.

7.28 atomic_timed_wait.h File Reference

7.28.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <atomic>.

7.29 atomic_wait.h File Reference

7.29.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <atomic>.

7.30 basic_ios.h File Reference

Classes

- class std::basic_ios< _CharT, _Traits >

Namespaces

- namespace std

Functions

- template<typename _Facet>
  const _Facet & **std::__check_facet** (const _Facet *__f)

7.30.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

7.31 basic_ios.tcc File Reference

Namespaces

- namespace std

7.31.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

7.32 basic_string.h File Reference

Classes

- class std::basic_string< _CharT, _Traits, _Alloc >
- struct std::hash< basic_string< char, char_traits< char >, _Alloc > >
- struct std::hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > >
- struct std::hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > >
- struct std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > >

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_TO_STRING_CONSTEXPR**

Functions

- template<typename _Str>
  constexpr _Str **std::__str_concat** (typename _Str::value_type const *__lhs, typename _Str::size_type __lhs_↩ len, typename _Str::value_type const *__rhs, typename _Str::size_type __rhs_len, typename _Str::allocator_type const &__a)
- constexpr void **std::__to_wstring_numeric** (const char *__s, int __len, wchar_t *__wout)
- wstring **std::__to_wstring_numeric** (const string &__s)
- template<typename _InputIterator, typename _CharT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_CharT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::basic_string (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> basic_string< _CharT, char_traits< _CharT >, _Allocator >
- template<typename _CharT, typename _Traits, typename _Allocator = allocator<_CharT>, typename = _RequireAllocator<_Allocator>>
  std::basic_string (basic_string_view< _CharT, _Traits >, const _Allocator &=_Allocator()) -> basic_string< ↩ _CharT, _Traits, _Allocator >

- template<typename _CharT, typename _Traits, typename _Allocator = allocator<_CharT>, typename = _RequireAllocator<_Allocator>>
  std::basic_string (basic_string_view< _CharT, _Traits >, typename basic_string< _CharT, _Traits, _Allocator >::size_type, typename basic_string< _CharT, _Traits, _Allocator >::size_type, const _Allocator &=_Allocator()) -> basic_string< _CharT, _Traits, _Allocator >
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &&__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)
- template<> basic_istream< char > & **std::getline** (basic_istream< char > &__in, basic_string< char > &__str, char __delim)
- template<> basic_istream< wchar_t > & **std::getline** (basic_istream< wchar_t > &__in, basic_string< wchar_t > &__str, wchar_t __delim)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (_CharT __lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (_CharT __lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (basic_string< _CharT, _Traits, _Alloc > &&__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (const _CharT *__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > **std::operator+** (const basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)

- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr basic_string< _CharT, _Traits, _Alloc > std::operator+ (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Alloc > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr auto std::operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr auto std::operator<=> (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr bool std::operator== (const basic_string< _CharT, _Traits, _Alloc > &__lhs, const basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)
- template<> basic_istream< char > & **std::operator**>> (basic_istream< char > &__is, basic_string< char > &__str)
- double **std::stod** (const string &__str, size_t *__idx=0)
- double **std::stod** (const wstring &__str, size_t *__idx=0)
- float **std::stof** (const string &__str, size_t *__idx=0)
- float **std::stof** (const wstring &__str, size_t *__idx=0)
- int **std::stoi** (const string &__str, size_t *__idx=0, int __base=10)
- int **std::stoi** (const wstring &__str, size_t *__idx=0, int __base=10)
- long **std::stol** (const string &__str, size_t *__idx=0, int __base=10)
- long **std::stol** (const wstring &__str, size_t *__idx=0, int __base=10)
- long double **std::stold** (const string &__str, size_t *__idx=0)
- long double **std::stold** (const wstring &__str, size_t *__idx=0)
- long long **std::stoll** (const string &__str, size_t *__idx=0, int __base=10)
- long long **std::stoll** (const wstring &__str, size_t *__idx=0, int __base=10)
- unsigned long **std::stoul** (const string &__str, size_t *__idx=0, int __base=10)
- unsigned long **std::stoul** (const wstring &__str, size_t *__idx=0, int __base=10)
- unsigned long long **std::stoull** (const string &__str, size_t *__idx=0, int __base=10)
- unsigned long long **std::stoull** (const wstring &__str, size_t *__idx=0, int __base=10)
- template<typename _CharT, typename _Traits, typename _Alloc>
  constexpr void std::swap (basic_string< _CharT, _Traits, _Alloc > &__lhs, basic_string< _CharT, _Traits, _Alloc > &__rhs) noexcept(/*conditional */)
- string **std::to_string** (double __val)
- string **std::to_string** (float __val)
- string **std::to_string** (int __val) noexcept
- string **std::to_string** (long __val) noexcept
- string **std::to_string** (long double __val)
- string **std::to_string** (long long __val)
- string **std::to_string** (unsigned __val) noexcept
- string **std::to_string** (unsigned long __val) noexcept
- string **std::to_string** (unsigned long long __val)
- wstring **std::to_wstring** (double __val)
- wstring **std::to_wstring** (float __val)

- wstring **std::to_wstring** (int __val)
- wstring **std::to_wstring** (long __val)
- wstring **std::to_wstring** (long double __val)
- wstring **std::to_wstring** (long long __val)
- wstring **std::to_wstring** (unsigned __val)
- wstring **std::to_wstring** (unsigned long __val)
- wstring **std::to_wstring** (unsigned long long __val)

7.32.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.

7.33 basic_string.tcc File Reference

Namespaces

- namespace std

Functions

- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Alloc>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Alloc > &__str)

7.33.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.

7.34 backward/binders.h File Reference

Classes

- class std::binder1st< _Operation >
- class std::binder2nd< _Operation >

Namespaces

- namespace std

Functions

- template<typename _Operation, typename _Tp>
  binder1st< _Operation > std::bind1st (const _Operation &__fn, const _Tp &__x)
- template<typename _Operation, typename _Tp>
  binder2nd< _Operation > std::bind2nd (const _Operation &__fn, const _Tp &__x)

7.34.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

7.35 bits/binders.h File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Fn, typename... _Args>
  using **std::_Bind_back_t**
- template<typename _Fn, typename... _Args>
  using **std::_Bind_front_t**

Functions

- template<typename... _BoundArgs, typename... _Args>
  constexpr auto **std::__make_bound_args** (_Args &&... __args)

7.35.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

7.36 boost_concept_check.h File Reference

Namespaces

- namespace __gnu_cxx
- namespace __gnu_debug
- namespace std

Macros

- #define **_GLIBCXX_CLASS_REQUIRES**(_type_var, _ns, _concept)
- #define **_GLIBCXX_CLASS_REQUIRES2**(_type_var1, _type_var2, _ns, _concept)
- #define **_GLIBCXX_CLASS_REQUIRES3**(_type_var1, _type_var2, _type_var3, _ns, _concept)
- #define **_GLIBCXX_CLASS_REQUIRES4**(_type_var1, _type_var2, _type_var3, _type_var4, _ns, _concept)
- #define **_GLIBCXX_DEFINE_BINARY_OPERATOR_CONSTRAINT**(_OP, _NAME)
- #define **_GLIBCXX_DEFINE_BINARY_PREDICATE_OP_CONSTRAINT**(_OP, _NAME)
- #define **_IsUnused**

Functions

- template<class _Tp>
  void **__gnu_cxx::__aux_require_boolean_expr** (const _Tp &__t)
- void **__gnu_cxx::__error_type_must_be_a_signed_integer_type** ()
- void **__gnu_cxx::__error_type_must_be_an_integer_type** ()
- void **__gnu_cxx::__error_type_must_be_an_unsigned_integer_type** ()
- template<class _Concept>
  constexpr void **__gnu_cxx::__function_requires** ()

7.36.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

7.37 c++0x_warning.h File Reference

7.37.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

7.38 char_traits.h File Reference

Classes

- struct __gnu_cxx::_Char_types< _CharT >
- struct __gnu_cxx::char_traits< _CharT >
- struct std::char_traits< _CharT >
- struct std::char_traits< char >
- struct std::char_traits< wchar_t >

Namespaces

- namespace __gnu_cxx
- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_ALWAYS_INLINE**

Functions

- template<typename _ChTraits>
  constexpr auto **std::__detail::__char_traits_cmp_cat** (int __cmp) noexcept

7.38.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.

7.39 charconv.h File Reference

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Tp>
  constexpr void **std::__detail::__to_chars_10_impl** (char *__first, unsigned __len, _Tp __val) noexcept
- template<typename _Tp>
  constexpr unsigned **std::__detail::__to_chars_len** (_Tp __value, int __base) noexcept

7.39.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <charconv>.

7.40 chrono.h File Reference

Classes

- struct std::common_type< chrono::duration< _Rep, _Period > >
- struct std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >
- struct std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >
- struct std::common_type< chrono::time_point< _Clock, _Duration > >
- struct std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >
- struct std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >
- class std::chrono::duration< _Rep, _Period >
- struct std::chrono::duration_values< _Rep >
- struct std::chrono::steady_clock
- struct std::chrono::system_clock
- class std::chrono::time_point< _Clock, _Dur >
- struct std::chrono::treat_as_floating_point< _Rep >

Namespaces

- namespace std
- namespace std::chrono
- namespace std::filesystem
- namespace std::literals
- namespace std::literals::chrono_literals

Typedefs

- using std::chrono::days
- using **std::chrono::file_clock**
- template<typename _Duration>
  using **std::chrono::file_time**
- using std::chrono::high_resolution_clock
- using std::chrono::hours
- using std::chrono::microseconds
- using std::chrono::milliseconds
- using std::chrono::minutes
- using std::chrono::months
- using std::chrono::nanoseconds
- using std::chrono::seconds
- using **std::chrono::sys_days**
- using **std::chrono::sys_seconds**
- template<typename _Duration>
  using **std::chrono::sys_time**
- using std::chrono::weeks
- using std::chrono::years

Functions

- template<typename _Rep, typename _Period>
  constexpr enable_if_t< numeric_limits< _Rep >::is_signed, duration< _Rep, _Period > > std::chrono::abs (duration< _Rep, _Period > __d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::__detail::ceil (const duration< _Rep, _Period > &↩ __d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::ceil (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::ceil (const time_point< _Clock, _Dur > &__tp)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::duration_cast (const duration< _Rep, _Period > &↩ __d)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr __enable_if_is_duration< _ToDur > std::chrono::floor (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur >, time_point< _Clock, _ToDur > > std::chrono::floor (const time_point< _Clock, _Dur > &__tp)
- template<char... _Digits>
  constexpr chrono::hours std::literals::chrono_literals::operator""h ()
- constexpr chrono::duration< long double, ratio< 3600, 1 > > std::literals::chrono_literals::operator""h (long double __hours)
- template<char... _Digits>
  constexpr chrono::minutes std::literals::chrono_literals::operator""min ()
- constexpr chrono::duration< long double, ratio< 60, 1 > > std::literals::chrono_literals::operator""min (long double __mins)
- template<char... _Digits>
  constexpr chrono::milliseconds std::literals::chrono_literals::operator""ms ()
- constexpr chrono::duration< long double, milli > std::literals::chrono_literals::operator""ms (long double __↩ msecs)
- template<char... _Digits>
  constexpr chrono::nanoseconds std::literals::chrono_literals::operator""ns ()
- constexpr chrono::duration< long double, nano > std::literals::chrono_literals::operator""ns (long double __nsecs)
- template<char... _Digits>
  constexpr chrono::seconds std::literals::chrono_literals::operator""s ()
- constexpr chrono::duration< long double > std::literals::chrono_literals::operator""s (long double __secs)
- template<char... _Digits>
  constexpr chrono::microseconds std::literals::chrono_literals::operator""us ()
- constexpr chrono::duration< long double, micro > std::literals::chrono_literals::operator""us (long double __↩ usecs)
- template<typename _ToDur, typename _Rep, typename _Period>
  constexpr enable_if_t< __and_< __is_duration< _ToDur >, __not_< treat_as_floating_point< typename _To↩ Dur::rep > > >::value, _ToDur > std::chrono::round (const duration< _Rep, _Period > &__d)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr enable_if_t< __is_duration_v< _ToDur > &&!treat_as_floating_point_v< typename _ToDur::rep >, time_point< _Clock, _ToDur > > std::chrono::round (const time_point< _Clock, _Dur > &__tp)
- template<typename _ToDur, typename _Clock, typename _Dur>
  constexpr __enable_if_t< __is_duration< _ToDur >::value, time_point< _Clock, _ToDur > > std::chrono::time_point_cast (const time_point< _Clock, _Dur > &__t)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator-
  (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::operator% (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< duration< _Rep1, _Period1 >, duration< _Rep2, _Period2 > >::type std::chrono::operator%
  (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Rep2, typename _Period>
  constexpr duration< __common_rep_t< _Rep2, _Rep1 >, _Period > std::chrono::operator* (const _Rep1 &__s, const duration< _Rep2, _Period > &__d)
- template<typename _Rep1, typename _Period, typename _Rep2>
  constexpr duration< __common_rep_t< _Rep1, __disable_if_is_duration< _Rep2 > >, _Period > std::chrono::operator/ (const duration< _Rep1, _Period > &__d, const _Rep2 &__s)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr common_type< _Rep1, _Rep2 >::type std::chrono::operator/ (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator< (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator<= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr auto std::chrono::operator<=> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator> (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)
- template<typename _Rep1, typename _Period1, typename _Rep2, typename _Period2>
  constexpr bool std::chrono::operator>= (const duration< _Rep1, _Period1 > &__lhs, const duration< _Rep2, _Period2 > &__rhs)

- template<typename _Rep1, typename _Period1, typename _Clock, typename _Dur2>
  constexpr time_point< _Clock, typename common_type< duration< _Rep1, _Period1 >, _Dur2 >::type > std::chrono::operator+ (const duration< _Rep1, _Period1 > &__lhs, const time_point< _Clock, _Dur2 > &↩
  __rhs)
- template<typename _Clock, typename _Dur1, typename _Rep2, typename _Period2>
  constexpr time_point< _Clock, typename common_type< _Dur1, duration< _Rep2, _Period2 > >::type > std::chrono::operator- (const time_point< _Clock, _Dur1 > &__lhs, const duration< _Rep2, _Period2 > &_↩
  _rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr common_type< _Dur1, _Dur2 >::type std::chrono::operator- (const time_point< _Clock, _Dur1 > &↩
  __lhs, const time_point< _Clock, _Dur2 > &__rhs)

- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator<** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator<=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, three_way_comparable_with< _Dur1 > _Dur2>
  constexpr auto **std::chrono::operator<=>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator>** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)
- template<typename _Clock, typename _Dur1, typename _Dur2>
  constexpr bool **std::chrono::operator>=** (const time_point< _Clock, _Dur1 > &__lhs, const time_point< _Clock, _Dur2 > &__rhs)

Variables

- template<typename _Tp>
  constexpr bool **std::chrono::is_clock_v**
- template<typename _Tp>
  constexpr bool **std::chrono::is_clock_v< _Tp >**
- template<> constexpr bool **std::chrono::is_clock_v< file_clock >**
- template<> constexpr bool **std::chrono::is_clock_v< steady_clock >**
- template<> constexpr bool **std::chrono::is_clock_v< system_clock >**
- template<typename _Rep>
  constexpr bool **std::chrono::treat_as_floating_point_v**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< double >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< float >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< int >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< long >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< long double >**
- template<> constexpr bool **std::chrono::treat_as_floating_point_v< long long >**

7.40.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <chrono>.

7.40.2 Function Documentation

ceil()

```
template<typename _ToDur, typename _Rep, typename _Period>
__enable_if_is_duration< _ToDur > std::chrono::ceil (
            const duration< _Rep, _Period > & __d)  [nodiscard], [constexpr]
```

Convert a `duration` to type `ToDur` and round up.
If the duration cannot be represented exactly in the result type, returns the closest value that is greater than the argument.

Template Parameters

_ToDur	The result type must be a `duration`.

Parameters

_↩ _d	A duration.

Returns

The value of __d converted to type _ToDur.

Since

C++17

7.41 chrono_io.h File Reference

Namespaces

- namespace std
- namespace std::chrono

Functions

- void **std::chrono::__detail::from_stream** ()=delete
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>> basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, day &__d, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Rep, typename _Period, typename _Alloc = allocator<_CharT>> basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, duration< _Rep, _Period > &__d, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>> basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, file_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>> basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, gps_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>> basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, local_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>> basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, month &__m, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *_↩ _offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>> basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, month_day &__md, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__↩ offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>> basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, sys_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)

- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, tai_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Duration, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, utc_time< _Duration > &__tp, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, weekday &__wd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, year &__y, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, year_month &__ym, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _CharT, typename _Traits, typename _Alloc = allocator<_CharT>>
basic_istream< _CharT, _Traits > & **std::chrono::from_stream** (basic_istream< _CharT, _Traits > &__is, const _CharT *__fmt, year_month_day &__ymd, basic_string< _CharT, _Traits, _Alloc > *__abbrev=nullptr, minutes *__offset=nullptr)
- template<typename _Duration>
__detail::__local_time_fmt< _Duration > std::chrono::local_time_format (local_time< _Duration > __time, const string *__abbrev=nullptr, const seconds *__offset_sec=nullptr)
- template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const day &__d)
- template<typename _CharT, typename _Traits, typename _Duration>
basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const file_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const gps_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const hh_mm_ss< _Duration > &__hms)
- template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const local_info &__li)
- template<typename _CharT, typename _Traits, typename _Duration>
requires requires(const sys_time<_Duration>& __st) { __os << __st; }
basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const local_time< _Duration > &__lt)
- template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month &__m)
- template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_day &__md)
- template<typename _CharT, typename _Traits>
basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_day_last &__mdl)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_weekday &__mwd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const month_weekday_last &__mwdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const sys_days &__dp)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::chrono::operator<< (basic_ostream< _CharT, _Traits > &__os, const sys_info &__i)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const sys_time< _Duration > &__tp)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const tai_time< _Duration > &__t)
- template<typename _CharT, typename _Traits, typename _Duration>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const utc_time< _Duration > &__t)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const weekday &__wd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const weekday_indexed &__wdi)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const weekday_last &__wdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year &__y)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month &__ym)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_day &__ymd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_day_last &__ymdl)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_weekday &__ymwd)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const year_month_weekday_last &__ymwdl)
- template<typename _CharT, typename _Traits, typename _Duration, typename _TimeZonePtr>
  basic_ostream< _CharT, _Traits > & **std::chrono::operator<<** (basic_ostream< _CharT, _Traits > &__os, const zoned_time< _Duration, _TimeZonePtr > &__t)
- template<typename _CharT, typename _Traits, typename _Rep, typename _Period>
  basic_ostream< _CharT, _Traits > & std::chrono::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const duration< _Rep, _Period > &__d)

- template<typename _CharT, __detail::__parsable< _CharT > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &_abbrev)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &_abbrev, minutes &__offset)
- template<typename _CharT, typename _Traits = char_traits<_CharT>, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const _CharT *__fmt, _Parsable &__tp, minutes &__offset)
- template<typename _CharT, typename _Traits, typename _Alloc, __detail::__parsable< _CharT, _Traits > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits, _Alloc>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, basic_string< _CharT, _Traits, _Alloc > &__abbrev, minutes &__offset)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _StrT = basic_string<_CharT, _Traits>, __detail::__parsable< _CharT, _Traits, _StrT, minutes > _Parsable>
  auto **std::chrono::parse** (const basic_string< _CharT, _Traits, _Alloc > &__fmt, _Parsable &__tp, minutes &_offset)

7.41.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <chrono>.

7.42 codecvt.h File Reference

Classes

- class std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >
- class std::codecvt< _InternT, _ExternT, _StateT >
- class std::codecvt< char, char, mbstate_t >
- class std::codecvt< char16_t, char, mbstate_t >
- class std::codecvt< char32_t, char, mbstate_t >
- class std::codecvt< wchar_t, char, mbstate_t >
- class std::codecvt_base
- class std::codecvt_byname< _InternT, _ExternT, _StateT >

Namespaces

- namespace std

Functions

- template bool **std::has_facet**< **codecvt**< **char, char, mbstate_t** > > (const locale &)
- template bool **std::has_facet**< **codecvt**< **wchar_t, char, mbstate_t** > > (const locale &)
- template const codecvt< char, char, mbstate_t > & **std::use_facet**< **codecvt**< **char, char, mbstate_t** > > (const locale &)
- template const codecvt< wchar_t, char, mbstate_t > & **std::use_facet**< **codecvt**< **wchar_t, char, mbstate_t** > > (const locale &)

Variables

- template<typename _InternT, typename _ExternT, typename _StateT>
  locale::id **std::codecvt**< **_InternT, _ExternT, _StateT** >**::id**

7.42.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.43 concept_check.h File Reference

Macros

- #define **__glibcxx_class_requires**(_a, _b)
- #define **__glibcxx_class_requires2**(_a, _b, _c)
- #define **__glibcxx_class_requires3**(_a, _b, _c, _d)
- #define **__glibcxx_class_requires4**(_a, _b, _c, _d, _e)
- #define **__glibcxx_function_requires**(...)

7.43.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

7.44 cow_string.h File Reference

Classes

- class std::basic_string< _CharT, _Traits, _Alloc >

Namespaces

- namespace std

Functions

- template<typename _Operation>
  void **std::basic_string**< **_CharT, _Traits, _Alloc** >**__resize_and_overwrite** (const size_type __n, _Operation __op)

Variables

- template<typename _CharT, typename _Traits, typename _Alloc>
  const basic_string< _CharT, _Traits, _Alloc >::size_type **std::basic_string**< **_CharT, _Traits, _Alloc** >**::npos**

7.44.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.
Defines the reference-counted COW string implementation.

7.45 cpp_type_traits.h File Reference

Namespaces

- namespace std

Macros

- #define **__INT_N**(TYPE)

Functions

- template<typename _Iterator>
  constexpr _Iterator **std::__miter_base** (_Iterator __it)

Variables

- template<typename _ValT, typename _Tp>
  constexpr bool **std::__can_use_memchr_for_find**

7.45.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/type_traits.h>.

7.46 cpyfunc_impl.h File Reference

Classes

- class std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_MOF_CV_REF**
- #define **_GLIBCXX_MOF_INV_QUALS**
- #define **_GLIBCXX_MOF_REF**

7.46.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

7.47 cxxabi_forced.h File Reference

Classes

- class __cxxabiv1::__forced_unwind

7.47.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <cxxabi.h>.

7.48 cxxabi_init_exception.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CDTOR_CALLABI**
- #define **_GLIBCXX_HAVE_CDTOR_CALLABI**

Functions

- void * **__cxxabiv1::__cxa_allocate_exception** (size_t) noexcept
- void **__cxxabiv1::__cxa_free_exception** (void *) noexcept
- __cxa_refcounted_exception * **__cxxabiv1::__cxa_init_primary_exception** (void *__object, std::type_info *↩ __tinfo, void(*__dest)(void *)) noexcept

7.48.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

7.49 deque.tcc File Reference

Namespaces

- namespace std

Functions

- template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **std::__copy_move_a1** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **std::__copy_move_a1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Tp>
  __gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value,::_Deque_iterator< _Tp, _Tp &, _Tp * > >::__type **std::__copy_move_a1** (_II __first, _II __last, ::_Deque_iterator< _Tp, _Tp &, _Tp * > __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value,::_Deque_iterator< _CharT, _CharT &, _CharT * > >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT, char_traits< _CharT > > __first, istreambuf_iterator< _CharT, char_traits< _CharT > > __last, ::_Deque_iterator< _CharT, _CharT &, _↩ CharT * > __result)
- template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **std::__copy_move_backward_a1** (::_Deque_iterator< _ITp, _↩ IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **std::__copy_move_backward_a1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)

- template<bool _IsMove, typename _II, typename _Tp>
  __gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value,::_Deque_iterator< _Tp, _Tp &, _Tp * > >::__type **std::__copy_move_backward_a1** (_II __first, _II __last, ::_Deque_iterator< _Tp, _Tp &, _Tp * > __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **std::__copy_move_backward_dit** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **std::__copy_move_dit** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value,::_Deque_iterator< _CharT, _CharT &, _CharT * > >::__type **std::__copy_n_a** (istreambuf_iterator< _CharT, char_traits< _CharT > > __it, _Size __size, ::_Deque_iterator< _CharT, _CharT &, _CharT * > __result, bool __strict)
- template<typename _Tp, typename _Ref, typename _Ptr, typename _II>
  __gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value, bool >::__type **std::__equal_aux1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **std::__equal_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)
- template<typename _II, typename _Tp, typename _Ref, typename _Ptr>
  __gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value, bool >::__type **std::__equal_aux1** (_II __first1, _II __last1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __first2)
- template<typename _Tp, typename _Ref, typename _Ptr, typename _II>
  bool **std::__equal_dit** (const ::_Deque_iterator< _Tp, _Ref, _Ptr > &__first1, const ::_Deque_iterator< _Tp, _Ref, _Ptr > &__last1, _II __first2)
- template<typename _Tp, typename _VTp>
  void **std::__fill_a1** (const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _VTp &__value)
- template<typename _Tp1, typename _Ref, typename _Ptr, typename _Tp2>
  int **std::__lex_cmp_dit** (::_Deque_iterator< _Tp1, _Ref, _Ptr > __first1, ::_Deque_iterator< _Tp1, _Ref, _Ptr > __last1, const _Tp2 *__first2, const _Tp2 *__last2)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **std::__lexicographical_compare_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2>
  bool **std::__lexicographical_compare_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, _Tp2 *__first2, _Tp2 *__last2)
- template<typename _Tp1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **std::__lexicographical_compare_aux1** (_Tp1 *__first1, _Tp1 *__last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)
- template<typename _ITp, typename _IRef, typename _IPtr, typename _OTp, typename _Tp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **std::__uninitialized_copy_a** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result, allocator< _Tp > &)
- template<typename _Iter, typename _OTp, typename _Tp>
  __enable_if_t< __is_random_access_iter< _Iter >::value, ::_Deque_iterator< _OTp, _OTp &, _OTp * > > **std::__uninitialized_copy_a** (_Iter __first, _Iter __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result, allocator< _Tp > &)
- template<typename _ITp, typename _IRef, typename _IPtr, typename _OTp, typename _Tp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **std::__uninitialized_move_a** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result, allocator< _Tp > &)

7.49.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <deque>.

7.50 enable_special_members.h File Reference

Namespaces

- namespace std

7.50.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

7.51 erase_if.h File Reference

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Container, typename _UnsafeContainer, typename _Predicate>
  constexpr _Container::size_type **std::__detail::__erase_if** (_Container &__cont, _UnsafeContainer &__ucont, _Predicate __pred)
- template<typename _Container, typename _UnsafeContainer, typename _Predicate>
  _Container::size_type **std::__detail::__erase_nodes_if** (_Container &__cont, _UnsafeContainer &__ucont, _↩
  Predicate __pred)

7.51.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

7.52 exception.h File Reference

Classes

- class std::exception

Namespaces

- namespace std

7.52.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

7.53 exception_defines.h File Reference

Macros

- #define **__catch**(X)
- #define **__throw_exception_again**
- #define **__try**

7.53.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <exception>.

7.54 exception_ptr.h File Reference

Classes

- class std::__unspecified__::exception_ptr
- class std::exception_ptr

Namespaces

- namespace std

Functions

- exception_ptr std::current_exception () noexcept
- template<typename _Ex>
  constexpr exception_ptr std::make_exception_ptr (_Ex __ex) noexcept
- void std::__unspecified__::rethrow_exception (exception_ptr)
- void std::rethrow_exception (exception_ptr)
- constexpr void **std::swap** (exception_ptr &__lhs, exception_ptr &__rhs)

7.54.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <exception>.

7.54.2 Function Documentation

rethrow_exception()

```
void std::rethrow_exception (
            exception_ptr )
```

Throw the object pointed to by the exception_ptr.

7.55 formatfwd.h File Reference

7.55.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <format>.

7.56 forward_list.h File Reference

Classes

- struct std::_Fwd_list_base< _Tp, _Alloc >
- struct std::_Fwd_list_const_iterator< _Tp >
- struct std::_Fwd_list_iterator< _Tp >
- struct std::_Fwd_list_node< _Tp >
- struct std::_Fwd_list_node_base
- class std::__fwdlist::_Iterator< _Const, _Ptr >
- struct std::__fwdlist::_Node< _ValPtr >

- struct std::__fwdlist::_Node_base< _VoidPtr >
- class std::forward_list< _Tp, _Alloc >

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_FWDLIST_REMOVE_RETURN_TYPE_TAG**
- #define **_GLIBCXX_USE_ALLOC_PTR_FOR_FWD_LIST**

Functions

- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::forward_list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward_list< _ValT, _Allocator >
- template<typename _Tp, typename _Alloc>
  __detail::__synth3way_t< _Tp > std::operator<=> (const forward_list< _Tp, _Alloc > &__x, const forward_list< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp, typename _Alloc>
  void std::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))

7.56.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <forward_list>.

7.57 forward_list.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX20_ONLY**(__expr)

Functions

- template<typename _Tp, typename _Alloc>
  bool std::operator== (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)

7.57.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <forward_list>.

7.58 bits/fs_dir.h File Reference

Classes

- class std::filesystem::directory_entry
- class std::filesystem::directory_iterator
- class std::filesystem::file_status
- class std::filesystem::recursive_directory_iterator

Namespaces

- namespace std
- namespace std::filesystem

Variables

- template<> constexpr bool **std::ranges::enable_borrowed_range**< **filesystem::directory_iterator** >
- template<> constexpr bool **std::ranges::enable_borrowed_range**< **filesystem::recursive_directory_iterator** >
- template<> constexpr bool **std::ranges::enable_view**< **filesystem::directory_iterator** >
- template<> constexpr bool **std::ranges::enable_view**< **filesystem::recursive_directory_iterator** >

7.58.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <filesystem>.

7.59 experimental/bits/fs_dir.h File Reference

Namespaces

- namespace std
- namespace std::experimental

Functions

- directory_iterator **std::experimental::filesystem::begin** (directory_iterator __iter) noexcept
- recursive_directory_iterator **std::experimental::filesystem::begin** (recursive_directory_iterator __iter) noexcept
- directory_iterator **std::experimental::filesystem::end** (directory_iterator) noexcept
- recursive_directory_iterator **std::experimental::filesystem::end** (recursive_directory_iterator) noexcept
- bool **std::experimental::filesystem::operator!=** (const directory_iterator &__lhs, const directory_iterator &__rhs)
- bool **std::experimental::filesystem::operator!=** (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs)
- bool **std::experimental::filesystem::operator==** (const directory_iterator &__lhs, const directory_iterator &__rhs)
- bool **std::experimental::filesystem::operator==** (const recursive_directory_iterator &__lhs, const recursive_directory_iterator &__rhs)

7.59.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <experimental/filesystem>.

7.60 bits/fs_fwd.h File Reference

Classes

- struct std::filesystem::space_info

Namespaces

- namespace std
- namespace std::filesystem

Typedefs

- using std::filesystem::file_time_type

Enumerations

- enum class std::filesystem::copy_options : unsigned short {
  none , **skip_existing** , **overwrite_existing** , **update_existing** ,
  recursive , **copy_symlinks** , **skip_symlinks** , **directories_only** ,
  create_symlinks , **create_hard_links** }
- enum class std::filesystem::directory_options : unsigned char { **none** , **follow_directory_symlink** , **skip_↩ permission_denied** }
- enum class std::filesystem::file_type : signed char {
  none , **not_found** , **regular** , **directory** ,
  symlink , **block** , **character** , **fifo** ,
  socket , **unknown** }
- enum class std::filesystem::perm_options : unsigned { **replace** , **add** , **remove** , **nofollow** }
- enum class std::filesystem::perms : unsigned {
  none , **owner_read** , **owner_write** , **owner_exec** ,
  owner_all , **group_read** , **group_write** , **group_exec** ,
  group_all , **others_read** , **others_write** , **others_exec** ,
  others_all , **all** , **set_uid** , **set_gid** ,
  sticky_bit , **mask** , **unknown** }

Functions

- void **std::filesystem::copy** (const path &__from, const path &__to, copy_options __options)
- void **std::filesystem::copy** (const path &__from, const path &__to, copy_options __options, error_code &)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option, error_code &)
- path **std::filesystem::current_path** ()
- bool **std::filesystem::exists** (file_status) noexcept
- uintmax_t **std::filesystem::file_size** (const path &)
- uintmax_t **std::filesystem::file_size** (const path &, error_code &) noexcept
- uintmax_t **std::filesystem::hard_link_count** (const path &)
- uintmax_t **std::filesystem::hard_link_count** (const path &, error_code &) noexcept
- bool **std::filesystem::is_other** (file_status) noexcept
- bool **std::filesystem::is_regular_file** (file_status) noexcept
- bool **std::filesystem::is_symlink** (file_status) noexcept
- file_time_type **std::filesystem::last_write_time** (const path &)
- file_time_type **std::filesystem::last_write_time** (const path &, error_code &) noexcept
- copy_options & **std::filesystem::operator&=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::filesystem::operator^** (copy_options __x, copy_options __y) noexcept

- copy_options & **std::filesystem::operator^=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::filesystem::operator|** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::filesystem::operator|=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::filesystem::operator~** (copy_options __x) noexcept
- void **std::filesystem::permissions** (const path &, perms, perm_options, error_code &) noexcept
- path **std::filesystem::proximate** (const path &__p, const path &__base, error_code &__ec)
- path **std::filesystem::relative** (const path &__p, const path &__base, error_code &__ec)
- bool **std::filesystem::remove** (const path &, error_code &) noexcept
- uintmax_t **std::filesystem::remove_all** (const path &)
- uintmax_t **std::filesystem::remove_all** (const path &, error_code &)
- file_status **std::filesystem::status** (const path &)
- file_status **std::filesystem::status** (const path &, error_code &) noexcept
- bool **std::filesystem::status_known** (file_status) noexcept
- file_status **std::filesystem::symlink_status** (const path &)
- file_status **std::filesystem::symlink_status** (const path &, error_code &) noexcept

- perms & **std::filesystem::operator&=** (perms &__x, perms __y) noexcept
- constexpr perms **std::filesystem::operator^** (perms __x, perms __y) noexcept
- perms & **std::filesystem::operator^=** (perms &__x, perms __y) noexcept
- constexpr perms **std::filesystem::operator|** (perms __x, perms __y) noexcept
- perms & **std::filesystem::operator|=** (perms &__x, perms __y) noexcept
- constexpr perms **std::filesystem::operator~** (perms __x) noexcept

- perm_options & **std::filesystem::operator&=** (perm_options &__x, perm_options __y) noexcept
- constexpr perm_options **std::filesystem::operator^** (perm_options __x, perm_options __y) noexcept
- perm_options & **std::filesystem::operator^=** (perm_options &__x, perm_options __y) noexcept
- constexpr perm_options **std::filesystem::operator|** (perm_options __x, perm_options __y) noexcept
- perm_options & **std::filesystem::operator|=** (perm_options &__x, perm_options __y) noexcept
- constexpr perm_options **std::filesystem::operator~** (perm_options __x) noexcept

- directory_options & **std::filesystem::operator&=** (directory_options &__x, directory_options __y) noexcept
- constexpr directory_options **std::filesystem::operator^** (directory_options __x, directory_options __y) noexcept
- directory_options & **std::filesystem::operator^=** (directory_options &__x, directory_options __y) noexcept
- constexpr directory_options **std::filesystem::operator|** (directory_options __x, directory_options __y) noexcept
- directory_options & **std::filesystem::operator|=** (directory_options &__x, directory_options __y) noexcept
- constexpr directory_options **std::filesystem::operator~** (directory_options __x) noexcept

7.60.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <filesystem>.

7.61 experimental/bits/fs_fwd.h File Reference

Classes

- struct std::experimental::filesystem::v1::space_info

Namespaces

- namespace std
- namespace std::experimental

Typedefs

- using std::experimental::filesystem::file_time_type

Enumerations

- enum class std::experimental::filesystem::copy_options : unsigned short {
  none , **skip_existing** , **overwrite_existing** , **update_existing** ,
  recursive , **copy_symlinks** , **skip_symlinks** , **directories_only** ,
  create_symlinks , **create_hard_links** }
- enum class std::experimental::filesystem::directory_options : unsigned char { **none** , **follow_directory_symlink** , **skip_permission_denied** }
- enum class std::experimental::filesystem::file_type : signed char {
  none , **not_found** , **regular** , **directory** ,
  symlink , **block** , **character** , **fifo** ,
  socket , **unknown** }
- enum class std::experimental::filesystem::perms : unsigned {
  none , **owner_read** , **owner_write** , **owner_exec** ,
  owner_all , **group_read** , **group_write** , **group_exec** ,
  group_all , **others_read** , **others_write** , **others_exec** ,
  others_all , **all** , **set_uid** , **set_gid** ,
  sticky_bit , **mask** , **unknown** , **add_perms** ,
  remove_perms , **symlink_nofollow** }

Functions

- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, copy_options __options)
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, copy_options __options, error_code &) noexcept
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option, error_code &)
- path **std::experimental::filesystem::current_path** ()
- bool **std::experimental::filesystem::is_regular_file** (file_status) noexcept
- bool **std::experimental::filesystem::is_symlink** (file_status) noexcept
- copy_options & **std::experimental::filesystem::operator&=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::experimental::filesystem::operator^** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::experimental::filesystem::operator^=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::experimental::filesystem::operator|** (copy_options __x, copy_options __y) noexcept
- copy_options & **std::experimental::filesystem::operator|=** (copy_options &__x, copy_options __y) noexcept
- constexpr copy_options **std::experimental::filesystem::operator~** (copy_options __x) noexcept
- file_status **std::experimental::filesystem::status** (const path &)
- file_status **std::experimental::filesystem::status** (const path &, error_code &) noexcept
- bool **std::experimental::filesystem::status_known** (file_status) noexcept
- file_status **std::experimental::filesystem::symlink_status** (const path &)
- file_status **std::experimental::filesystem::symlink_status** (const path &, error_code &) noexcept

- perms & **std::experimental::filesystem::operator&=** (perms &__x, perms __y) noexcept
- constexpr perms **std::experimental::filesystem::operator^** (perms __x, perms __y) noexcept
- perms & **std::experimental::filesystem::operator^=** (perms &__x, perms __y) noexcept
- constexpr perms **std::experimental::filesystem::operator|** (perms __x, perms __y) noexcept
- perms & **std::experimental::filesystem::operator|=** (perms &__x, perms __y) noexcept
- constexpr perms **std::experimental::filesystem::operator~** (perms __x) noexcept

- directory_options & **std::experimental::filesystem::operator&=** (directory_options &__x, directory_options _↩ _y) noexcept
- constexpr directory_options **std::experimental::filesystem::operator^** (directory_options __x, directory_↩ options __y) noexcept
- directory_options & **std::experimental::filesystem::operator^=** (directory_options &__x, directory_options _↩ _y) noexcept
- constexpr directory_options **std::experimental::filesystem::operator|** (directory_options __x, directory_options __y) noexcept
- directory_options & **std::experimental::filesystem::operator|=** (directory_options &__x, directory_options __y) noexcept
- constexpr directory_options **std::experimental::filesystem::operator~** (directory_options __x) noexcept

7.61.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <experimental/filesystem>.

7.62 bits/fs_ops.h File Reference

Namespaces

- namespace std
- namespace std::filesystem

Functions

- path **std::filesystem::absolute** (const path &__p)
- path **std::filesystem::absolute** (const path &__p, error_code &__ec)
- path **std::filesystem::canonical** (const path &__p)
- path **std::filesystem::canonical** (const path &__p, error_code &__ec)
- void **std::filesystem::copy** (const path &__from, const path &__to)
- void **std::filesystem::copy** (const path &__from, const path &__to, copy_options __options)
- void **std::filesystem::copy** (const path &__from, const path &__to, copy_options __options, error_code &)
- void **std::filesystem::copy** (const path &__from, const path &__to, error_code &__ec)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option, error_code &)
- bool **std::filesystem::copy_file** (const path &__from, const path &__to, error_code &__ec)
- void **std::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_symlink)
- void **std::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_symlink, error_code &__ec) noexcept
- bool **std::filesystem::create_directories** (const path &__p)
- bool **std::filesystem::create_directories** (const path &__p, error_code &__ec)
- bool **std::filesystem::create_directory** (const path &__p)

- bool **std::filesystem::create_directory** (const path &__p, const path &__attributes)
- bool **std::filesystem::create_directory** (const path &__p, const path &__attributes, error_code &__ec) noexcept
- bool **std::filesystem::create_directory** (const path &__p, error_code &__ec) noexcept
- void **std::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink)
- void **std::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- void **std::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link)
- void **std::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link, error_code &__ec) noexcept
- void **std::filesystem::create_symlink** (const path &__to, const path &__new_symlink)
- void **std::filesystem::create_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- path **std::filesystem::current_path** ()
- void **std::filesystem::current_path** (const path &__p)
- void **std::filesystem::current_path** (const path &__p, error_code &__ec) noexcept
- path **std::filesystem::current_path** (error_code &__ec)
- bool **std::filesystem::equivalent** (const path &__p1, const path &__p2)
- bool **std::filesystem::equivalent** (const path &__p1, const path &__p2, error_code &__ec) noexcept
- bool **std::filesystem::exists** (const path &__p)
- bool **std::filesystem::exists** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::exists** (file_status) noexcept
- uintmax_t **std::filesystem::file_size** (const path &)
- uintmax_t **std::filesystem::file_size** (const path &, error_code &) noexcept
- uintmax_t **std::filesystem::hard_link_count** (const path &)
- uintmax_t **std::filesystem::hard_link_count** (const path &, error_code &) noexcept
- bool **std::filesystem::is_block_file** (const path &__p)
- bool **std::filesystem::is_block_file** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_block_file** (file_status __s) noexcept
- bool **std::filesystem::is_character_file** (const path &__p)
- bool **std::filesystem::is_character_file** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_character_file** (file_status __s) noexcept
- bool **std::filesystem::is_directory** (const path &__p)
- bool **std::filesystem::is_directory** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_directory** (file_status __s) noexcept
- bool **std::filesystem::is_empty** (const path &__p)
- bool **std::filesystem::is_empty** (const path &__p, error_code &__ec)
- bool **std::filesystem::is_fifo** (const path &__p)
- bool **std::filesystem::is_fifo** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_fifo** (file_status __s) noexcept
- bool **std::filesystem::is_other** (const path &__p)
- bool **std::filesystem::is_other** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_other** (file_status) noexcept
- bool **std::filesystem::is_regular_file** (const path &__p)
- bool **std::filesystem::is_regular_file** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_regular_file** (file_status) noexcept
- bool **std::filesystem::is_socket** (const path &__p)
- bool **std::filesystem::is_socket** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_socket** (file_status __s) noexcept
- bool **std::filesystem::is_symlink** (const path &__p)
- bool **std::filesystem::is_symlink** (const path &__p, error_code &__ec) noexcept
- bool **std::filesystem::is_symlink** (file_status) noexcept

- file_time_type **std::filesystem::last_write_time** (const path &)
- file_time_type **std::filesystem::last_write_time** (const path &, error_code &) noexcept
- void **std::filesystem::last_write_time** (const path &__p, file_time_type __new_time)
- void **std::filesystem::last_write_time** (const path &__p, file_time_type __new_time, error_code &__ec) noexcept
- void **std::filesystem::permissions** (const path &, perms, perm_options, error_code &) noexcept
- void **std::filesystem::permissions** (const path &__p, perms __prms, error_code &__ec) noexcept
- void **std::filesystem::permissions** (const path &__p, perms __prms, perm_options __opts=perm_options::↩ replace)
- path **std::filesystem::proximate** (const path &__p, const path &__base, error_code &__ec)
- path **std::filesystem::proximate** (const path &__p, const path &__base=current_path())
- path **std::filesystem::proximate** (const path &__p, error_code &__ec)
- path **std::filesystem::read_symlink** (const path &__p)
- path **std::filesystem::read_symlink** (const path &__p, error_code &__ec)
- path **std::filesystem::relative** (const path &__p, const path &__base, error_code &__ec)
- path **std::filesystem::relative** (const path &__p, const path &__base=current_path())
- path **std::filesystem::relative** (const path &__p, error_code &__ec)
- bool **std::filesystem::remove** (const path &, error_code &) noexcept
- bool **std::filesystem::remove** (const path &__p)
- uintmax_t **std::filesystem::remove_all** (const path &)
- uintmax_t **std::filesystem::remove_all** (const path &, error_code &)
- void **std::filesystem::rename** (const path &__from, const path &__to)
- void **std::filesystem::rename** (const path &__from, const path &__to, error_code &__ec) noexcept
- void **std::filesystem::resize_file** (const path &__p, uintmax_t __size)
- void **std::filesystem::resize_file** (const path &__p, uintmax_t __size, error_code &__ec) noexcept
- space_info **std::filesystem::space** (const path &__p)
- space_info **std::filesystem::space** (const path &__p, error_code &__ec) noexcept
- file_status **std::filesystem::status** (const path &)
- file_status **std::filesystem::status** (const path &, error_code &) noexcept
- bool **std::filesystem::status_known** (file_status) noexcept
- file_status **std::filesystem::symlink_status** (const path &)
- file_status **std::filesystem::symlink_status** (const path &, error_code &) noexcept
- path **std::filesystem::temp_directory_path** ()
- path **std::filesystem::temp_directory_path** (error_code &__ec)
- path **std::filesystem::weakly_canonical** (const path &__p)
- path **std::filesystem::weakly_canonical** (const path &__p, error_code &__ec)

7.62.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <filesystem>.

7.63 experimental/bits/fs_ops.h File Reference

Namespaces

- namespace std
- namespace std::experimental

Functions

- path **std::experimental::filesystem::absolute** (const path &__p, const path &__base=current_path())
- path **std::experimental::filesystem::canonical** (const path &__p, const path &__base, error_code &__ec)
- path **std::experimental::filesystem::canonical** (const path &__p, const path &__base=current_path())
- path **std::experimental::filesystem::canonical** (const path &__p, error_code &__ec)
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to)
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, copy_options __options)
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, copy_options __options, error_code &__ec) noexcept
- void **std::experimental::filesystem::copy** (const path &__from, const path &__to, error_code &__ec) noexcept
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to)
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option)
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, copy_options __option, error_code &__ec)
- bool **std::experimental::filesystem::copy_file** (const path &__from, const path &__to, error_code &__ec)
- void **std::experimental::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_↩ symlink)
- void **std::experimental::filesystem::copy_symlink** (const path &__existing_symlink, const path &__new_↩ symlink, error_code &__ec) noexcept
- bool **std::experimental::filesystem::create_directories** (const path &__p)
- bool **std::experimental::filesystem::create_directories** (const path &__p, error_code &__ec)
- bool **std::experimental::filesystem::create_directory** (const path &__p)
- bool **std::experimental::filesystem::create_directory** (const path &__p, const path &__attributes)
- bool **std::experimental::filesystem::create_directory** (const path &__p, const path &__attributes, error_code &__ec) noexcept
- bool **std::experimental::filesystem::create_directory** (const path &__p, error_code &__ec) noexcept
- void **std::experimental::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink)
- void **std::experimental::filesystem::create_directory_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- void **std::experimental::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link)
- void **std::experimental::filesystem::create_hard_link** (const path &__to, const path &__new_hard_link, error_code &__ec) noexcept
- void **std::experimental::filesystem::create_symlink** (const path &__to, const path &__new_symlink)
- void **std::experimental::filesystem::create_symlink** (const path &__to, const path &__new_symlink, error_code &__ec) noexcept
- path **std::experimental::filesystem::current_path** ()
- void **std::experimental::filesystem::current_path** (const path &__p)
- void **std::experimental::filesystem::current_path** (const path &__p, error_code &__ec) noexcept
- path **std::experimental::filesystem::current_path** (error_code &__ec)
- bool **std::experimental::filesystem::equivalent** (const path &__p1, const path &__p2)
- bool **std::experimental::filesystem::equivalent** (const path &__p1, const path &__p2, error_code &__ec) noexcept
- bool **std::experimental::filesystem::exists** (const path &__p)
- bool **std::experimental::filesystem::exists** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::exists** (file_status __s) noexcept
- uintmax_t **std::experimental::filesystem::file_size** (const path &__p)
- uintmax_t **std::experimental::filesystem::file_size** (const path &__p, error_code &__ec) noexcept
- uintmax_t **std::experimental::filesystem::hard_link_count** (const path &__p)
- uintmax_t **std::experimental::filesystem::hard_link_count** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_block_file** (const path &__p)
- bool **std::experimental::filesystem::is_block_file** (const path &__p, error_code &__ec) noexcept

- bool **std::experimental::filesystem::is_block_file** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_character_file** (const path &__p)
- bool **std::experimental::filesystem::is_character_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_character_file** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_directory** (const path &__p)
- bool **std::experimental::filesystem::is_directory** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_directory** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_empty** (const path &__p)
- bool **std::experimental::filesystem::is_empty** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_fifo** (const path &__p)
- bool **std::experimental::filesystem::is_fifo** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_fifo** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_other** (const path &__p)
- bool **std::experimental::filesystem::is_other** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_other** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_regular_file** (const path &__p)
- bool **std::experimental::filesystem::is_regular_file** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_regular_file** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_socket** (const path &__p)
- bool **std::experimental::filesystem::is_socket** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_socket** (file_status __s) noexcept
- bool **std::experimental::filesystem::is_symlink** (const path &__p)
- bool **std::experimental::filesystem::is_symlink** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::is_symlink** (file_status __s) noexcept
- file_time_type **std::experimental::filesystem::last_write_time** (const path &__p)
- file_time_type **std::experimental::filesystem::last_write_time** (const path &__p, error_code &__ec) noexcept
- void **std::experimental::filesystem::last_write_time** (const path &__p, file_time_type __new_time)
- void **std::experimental::filesystem::last_write_time** (const path &__p, file_time_type __new_time, error_code &__ec) noexcept
- void **std::experimental::filesystem::permissions** (const path &__p, perms __prms)
- void **std::experimental::filesystem::permissions** (const path &__p, perms __prms, error_code &__ec) noexcept
- path **std::experimental::filesystem::read_symlink** (const path &__p)
- path **std::experimental::filesystem::read_symlink** (const path &__p, error_code &__ec)
- bool **std::experimental::filesystem::remove** (const path &__p)
- bool **std::experimental::filesystem::remove** (const path &__p, error_code &__ec) noexcept
- uintmax_t **std::experimental::filesystem::remove_all** (const path &__p)
- uintmax_t **std::experimental::filesystem::remove_all** (const path &__p, error_code &__ec)
- void **std::experimental::filesystem::rename** (const path &__from, const path &__to)
- void **std::experimental::filesystem::rename** (const path &__from, const path &__to, error_code &__ec) noexcept
- void **std::experimental::filesystem::resize_file** (const path &__p, uintmax_t __size)
- void **std::experimental::filesystem::resize_file** (const path &__p, uintmax_t __size, error_code &__ec) noexcept
- space_info **std::experimental::filesystem::space** (const path &__p)
- space_info **std::experimental::filesystem::space** (const path &__p, error_code &__ec) noexcept
- file_status **std::experimental::filesystem::status** (const path &__p)
- file_status **std::experimental::filesystem::status** (const path &__p, error_code &__ec) noexcept
- bool **std::experimental::filesystem::status_known** (file_status __s) noexcept
- file_status **std::experimental::filesystem::symlink_status** (const path &__p)
- file_status **std::experimental::filesystem::symlink_status** (const path &__p, error_code &__ec) noexcept

- path **std::experimental::filesystem::system_complete** (const path &__p)
- path **std::experimental::filesystem::system_complete** (const path &__p, error_code &__ec)
- path **std::experimental::filesystem::temp_directory_path** ()
- path **std::experimental::filesystem::temp_directory_path** (error_code &__ec)

7.63.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <experimental/filesystem>.

7.64 bits/fs_path.h File Reference

Classes

- class std::filesystem::filesystem_error
- class std::filesystem::path::iterator
- class std::filesystem::path

Namespaces

- namespace std
- namespace std::filesystem

Functions

- size_t **std::filesystem::hash_value** (const path &__p) noexcept

7.64.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <filesystem>.

7.65 experimental/bits/fs_path.h File Reference

Classes

- class std::experimental::filesystem::v1::basic_string_view< _CharT, _Traits >
- class std::experimental::filesystem::v1::filesystem_error
- class std::experimental::filesystem::v1::path::iterator
- class std::experimental::filesystem::v1::path

Namespaces

- namespace std
- namespace std::experimental

Functions

- bool **std::experimental::filesystem::operator<** (const path &__lhs, const path &__rhs) noexcept
- bool **std::experimental::filesystem::operator==** (const path &__lhs, const path &__rhs) noexcept

7.65.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <experimental/filesystem>.

7.66 fstream.tcc File Reference

Namespaces

- namespace std

7.66.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <fstream>.

7.67 funcref_impl.h File Reference

Classes

- class std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

Namespaces

- namespace std

7.67.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

7.68 functexcept.h File Reference

Namespaces

- namespace std

Functions

- void **std::__throw_bad_cast** (void)
- void **std::__throw_bad_exception** (void)
- void **std::__throw_bad_function_call** ()
- void **std::__throw_bad_typeid** (void)
- void **std::__throw_future_error** (int)
- void **std::__throw_ios_failure** (const char *)
- void **std::__throw_ios_failure** (const char *, int)
- void **std::__throw_range_error** (const char *)
- void **std::__throw_system_error** (int)

7.68.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <exception>.
This header provides support for -fno-exceptions.

7.69 functional_hash.h File Reference

Classes

- struct std::__is_fast_hash< _Hash >
- struct std::hash< _Tp >

- struct std::hash< _Tp * >
- struct std::hash< bool >
- struct std::hash< char >
- struct std::hash< char16_t >
- struct std::hash< char32_t >
- struct std::hash< double >
- struct std::hash< float >
- struct std::hash< int >
- struct std::hash< long >
- struct std::hash< long double >
- struct std::hash< long long >
- struct std::hash< short >
- struct std::hash< signed char >
- struct std::hash< unsigned char >
- struct std::hash< unsigned int >
- struct std::hash< unsigned long >
- struct std::hash< unsigned long long >
- struct std::hash< unsigned short >
- struct std::hash< wchar_t >

Namespaces

- namespace std

Macros

- #define **_Cxx_hashtable_define_trivial_hash**(_Tp)

Variables

- template<typename _Tp, typename = void>
  constexpr bool **std::__is_hash_enabled_for**
- template<typename _Tp>
  constexpr bool **std::__is_hash_enabled_for**< **_Tp, __void_t**< **decltype(hash**< **_Tp** >**()(declval**< **_Tp** >**()))**> >

7.69.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

7.70 funcwrap.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_MOF_CV**

Functions

- template<typename _Fn>
  std::function_ref (_Fn *) -> function_ref< _Fn >
- template<auto __fn, typename _Fn>
  std::function_ref (constant_wrapper< __fn, _Fn >) -> function_ref< remove_pointer_t< _Fn > >
- template<auto __fn, typename _Fn, typename _Tp, typename _SignaturePtr = decltype(__polyfunc::__deduce_funcref<_Fn, _Tp&>())>
  std::function_ref (constant_wrapper< __fn, _Fn >, _Tp &&) -> function_ref< remove_pointer_t< _Signature↩
  Ptr > >

7.70.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

7.71 gslice.h File Reference

Classes

- class std::gslice

Namespaces

- namespace std

7.71.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

7.72 gslice_array.h File Reference

Classes

- class std::gslice_array< _Tp >

Namespaces

- namespace std

7.72.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

7.73 hash_bytes.h File Reference

Namespaces

- namespace std

Functions

- size_t **std::_Fnv_hash_bytes** (const void *__ptr, size_t __len, size_t __seed)
- size_t **std::_Hash_bytes** (const void *__ptr, size_t __len, size_t __seed)

7.73.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

7.74 backward/hashtable.h File Reference

Namespaces

- namespace __gnu_cxx

Enumerations

- enum { **_S_num_primes** }

Functions

- unsigned long **__gnu_cxx::__stl_next_prime** (unsigned long __n)
- template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>
  bool **__gnu_cxx::operator!=** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<class _Val, class _Key, class _HF, class _Ex, class _Eq, class _All>
  bool **__gnu_cxx::operator==** (const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht1, const hashtable< _Val, _Key, _HF, _Ex, _Eq, _All > &__ht2)
- template<class _Val, class _Key, class _HF, class _Extract, class _EqKey, class _All>
  void **__gnu_cxx::swap** (hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht1, hashtable< _Val, _Key, _HF, _Extract, _EqKey, _All > &__ht2)

Variables

- template<typename _PrimeType>
  const _PrimeType **__gnu_cxx::_Hashtable_prime_list< _PrimeType >::__stl_prime_list** [_S_num_primes]

7.74.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

7.75 bits/hashtable.h File Reference

Namespaces

- namespace std

7.75.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <unordered_map> or < unordered_set>.

7.76 hashtable_policy.h File Reference

Namespaces

- namespace std

7.76.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <unordered_map> or <unordered_set>.

7.77 indirect.h File Reference

7.77.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.78 indirect_array.h File Reference

Classes

- class std::indirect_array< _Tp >

Namespaces

- namespace std

7.78.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

7.79 inplace_tags.h File Reference

Namespaces

- namespace std

Variables

- template<typename>
  constexpr bool **std::__is_in_place_index_v**
- template<size_t _Nm>
  constexpr bool **std::__is_in_place_index_v**< **in_place_index_t**< **_Nm** > >
- template<typename>
  constexpr bool **std::__is_in_place_type_v**
- template<typename _Tp>
  constexpr bool **std::__is_in_place_type_v**< **in_place_type_t**< **_Tp** > >
- constexpr in_place_t **std::in_place**
- template<size_t _Idx>
  constexpr in_place_index_t< _Idx > **std::in_place_index**
- template<typename _Tp>
  constexpr in_place_type_t< _Tp > **std::in_place_type**

7.79.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.
This file contains the parts of <utility> needed by other headers, so they don't need to include the whole of <utility>.

7.80 intcmp.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Tp, typename _Up>
  constexpr bool **std::cmp_equal** (_Tp __t, _Up __u) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **std::cmp_greater** (_Tp __t, _Up __u) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **std::cmp_greater_equal** (_Tp __t, _Up __u) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **std::cmp_less** (_Tp __t, _Up __u) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **std::cmp_less_equal** (_Tp __t, _Up __u) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **std::cmp_not_equal** (_Tp __t, _Up __u) noexcept
- template<typename _Res, typename _Tp>
  constexpr bool **std::in_range** (_Tp __t) noexcept

7.80.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.

7.81 invoke.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Tp, typename _Up = typename __inv_unwrap<_Tp>::type>
  constexpr _Up && **std::__invfwd** (typename remove_reference< _Tp >::type &__t) noexcept
- template<typename _Callable, typename... _Args>
  constexpr __invoke_result< _Callable, _Args... >::type std::__invoke (_Callable &&__fn, _Args &&... __args) noexcept(__is_nothrow_invocable< _Callable, _Args... >::value)
- template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>
  constexpr _Res **std::__invoke_impl** (__invoke_memfun_deref, _MemFun &&__f, _Tp &&__t, _Args &&... __↩
  args)
- template<typename _Res, typename _MemFun, typename _Tp, typename... _Args>
  constexpr _Res **std::__invoke_impl** (__invoke_memfun_ref, _MemFun &&__f, _Tp &&__t, _Args &&... __args)
- template<typename _Res, typename _MemPtr, typename _Tp>
  constexpr _Res **std::__invoke_impl** (__invoke_memobj_deref, _MemPtr &&__f, _Tp &&__t)
- template<typename _Res, typename _MemPtr, typename _Tp>
  constexpr _Res **std::__invoke_impl** (__invoke_memobj_ref, _MemPtr &&__f, _Tp &&__t)
- template<typename _Res, typename _Fn, typename... _Args>
  constexpr _Res **std::__invoke_impl** (__invoke_other, _Fn &&__f, _Args &&... __args)
- template<typename _Res, typename _Callable, typename... _Args>
  constexpr enable_if_t< is_invocable_r_v< _Res, _Callable, _Args... >, _Res > **std::__invoke_r** (_Callable &&__fn, _Args &&... __args) noexcept(is_nothrow_invocable_r_v< _Res, _Callable, _Args... >)

7.81.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

7.82 ios_base.h File Reference

Classes

- class std::ios_base::failure
- class std::ios_base

Namespaces

- namespace std

Macros

- #define **_NOREPLACE_UNUSED**

Enumerations

- enum **_Ios_Fmtflags** {
  _S_boolalpha , **_S_dec** , **_S_fixed** , **_S_hex** ,
  _S_internal , **_S_left** , **_S_oct** , **_S_right** ,
  _S_scientific , **_S_showbase** , **_S_showpoint** , **_S_showpos** ,
  _S_skipws , **_S_unitbuf** , **_S_uppercase** , **_S_adjustfield** ,
  _S_basefield , **_S_floatfield** , **_S_ios_fmtflags_end** , **_S_ios_fmtflags_max** ,
  _S_ios_fmtflags_min }
- enum **_Ios_Iostate** {
  _S_goodbit , **_S_badbit** , **_S_eofbit** , **_S_failbit** ,
  _S_ios_iostate_end , **_S_ios_iostate_max** , **_S_ios_iostate_min** }
- enum **_Ios_Openmode** {
  _S_app , **_S_ate** , **_S_bin** , **_S_in** ,
  _S_out , **_S_trunc** , **_S_noreplace** , **_S_ios_openmode_end** ,
  _S_ios_openmode_max , **_S_ios_openmode_min** }
- enum **_Ios_Seekdir** { **_S_beg** , **_S_cur** , **_S_end** , **_S_ios_seekdir_end** }
- enum class std::io_errc { **stream** }

Functions

- ios_base & std::boolalpha (ios_base &__base)
- ios_base & std::dec (ios_base &__base)
- ios_base & std::defaultfloat (ios_base &__base)
- ios_base & std::fixed (ios_base &__base)
- ios_base & std::hex (ios_base &__base)
- ios_base & std::hexfloat (ios_base &__base)
- ios_base & std::internal (ios_base &__base)
- const error_category & **std::iostream_category** () noexcept
- ios_base & std::left (ios_base &__base)
- error_code **std::make_error_code** (io_errc __e) noexcept
- error_condition **std::make_error_condition** (io_errc __e) noexcept
- ios_base & std::noboolalpha (ios_base &__base)
- ios_base & std::noshowbase (ios_base &__base)

- ios_base & std::noshowpoint (ios_base &__base)
- ios_base & std::noshowpos (ios_base &__base)
- ios_base & std::noskipws (ios_base &__base)
- ios_base & std::nounitbuf (ios_base &__base)
- ios_base & std::nouppercase (ios_base &__base)
- ios_base & std::oct (ios_base &__base)
- constexpr _Ios_Fmtflags **std::operator&** (_Ios_Fmtflags __a, _Ios_Fmtflags __b) noexcept
- constexpr _Ios_Iostate **std::operator&** (_Ios_Iostate __a, _Ios_Iostate __b) noexcept
- constexpr _Ios_Openmode **std::operator&** (_Ios_Openmode __a, _Ios_Openmode __b) noexcept
- constexpr const _Ios_Fmtflags & **std::operator&=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b) noexcept
- constexpr const _Ios_Iostate & **std::operator&=** (_Ios_Iostate &__a, _Ios_Iostate __b) noexcept
- constexpr const _Ios_Openmode & **std::operator&=** (_Ios_Openmode &__a, _Ios_Openmode __b) noexcept
- constexpr _Ios_Fmtflags **std::operator^** (_Ios_Fmtflags __a, _Ios_Fmtflags __b) noexcept
- constexpr _Ios_Iostate **std::operator^** (_Ios_Iostate __a, _Ios_Iostate __b) noexcept
- constexpr _Ios_Openmode **std::operator^** (_Ios_Openmode __a, _Ios_Openmode __b) noexcept
- constexpr const _Ios_Fmtflags & **std::operator^=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b) noexcept
- constexpr const _Ios_Iostate & **std::operator^=** (_Ios_Iostate &__a, _Ios_Iostate __b) noexcept
- constexpr const _Ios_Openmode & **std::operator^=** (_Ios_Openmode &__a, _Ios_Openmode __b) noexcept
- constexpr _Ios_Fmtflags **std::operator|** (_Ios_Fmtflags __a, _Ios_Fmtflags __b) noexcept
- constexpr _Ios_Iostate **std::operator|** (_Ios_Iostate __a, _Ios_Iostate __b) noexcept
- constexpr _Ios_Openmode **std::operator|** (_Ios_Openmode __a, _Ios_Openmode __b) noexcept
- constexpr const _Ios_Fmtflags & **std::operator|=** (_Ios_Fmtflags &__a, _Ios_Fmtflags __b) noexcept
- constexpr const _Ios_Iostate & **std::operator|=** (_Ios_Iostate &__a, _Ios_Iostate __b) noexcept
- constexpr const _Ios_Openmode & **std::operator|=** (_Ios_Openmode &__a, _Ios_Openmode __b) noexcept
- constexpr _Ios_Fmtflags **std::operator~** (_Ios_Fmtflags __a) noexcept
- constexpr _Ios_Iostate **std::operator~** (_Ios_Iostate __a) noexcept
- constexpr _Ios_Openmode **std::operator~** (_Ios_Openmode __a) noexcept
- ios_base & std::right (ios_base &__base)
- ios_base & std::scientific (ios_base &__base)
- ios_base & std::showbase (ios_base &__base)
- ios_base & std::showpoint (ios_base &__base)
- ios_base & std::showpos (ios_base &__base)
- ios_base & std::skipws (ios_base &__base)
- ios_base & std::unitbuf (ios_base &__base)
- ios_base & std::uppercase (ios_base &__base)

7.82.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

7.83 iosfwd_file.h File Reference

Namespaces

- namespace std

Typedefs

- typedef basic_filebuf< char > std::filebuf
- typedef basic_fstream< char > std::fstream
- typedef basic_ifstream< char > std::ifstream
- typedef basic_ofstream< char > std::ofstream
- typedef basic_filebuf< wchar_t > std::wfilebuf
- typedef basic_fstream< wchar_t > std::wfstream
- typedef basic_ifstream< wchar_t > std::wifstream
- typedef basic_ofstream< wchar_t > std::wofstream

7.83.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

7.84 iosfwd_span.h File Reference

7.84.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

7.85 iosfwd_string.h File Reference

Namespaces

- namespace std

Typedefs

- typedef basic_istringstream< char > std::istringstream
- typedef basic_ostringstream< char > std::ostringstream
- typedef basic_stringbuf< char > std::stringbuf
- typedef basic_stringstream< char > std::stringstream
- typedef basic_istringstream< wchar_t > std::wistringstream
- typedef basic_ostringstream< wchar_t > std::wostringstream
- typedef basic_stringbuf< wchar_t > std::wstringbuf
- typedef basic_stringstream< wchar_t > std::wstringstream

7.85.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

7.86 iosfwd_sync.h File Reference

7.86.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

7.87 istream.tcc File Reference

Namespaces

- namespace std

Functions

- template<typename _CharT, typename _Traits>
  void **std::__istream_extract** (basic_istream< _CharT, _Traits > &, _CharT *, streamsize)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)

7.87.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <istream>.

7.88 iterator_concepts.h File Reference

Classes

- struct std::default_sentinel_t

Namespaces

- namespace std
- namespace std::__detail

Concepts

- concept std::indirectly_comparable
- concept std::indirectly_copyable
- concept std::indirectly_movable
- concept std::indirectly_readable
- concept std::indirectly_swappable
- concept std::indirectly_writable
- concept std::mergeable
- concept std::permutable
- concept std::sortable
- concept std::weakly_incrementable

Typedefs

- template<typename _Tp>
  using **std::__indirect_value_t**
- template<typename _Iter>
  using **std::__detail::__iter_concept**
- template<typename _Tp>
  using **std::__detail::__iter_diff_t**
- template<typename _Iter, typename _Tp = _Iter>
  using **std::__detail::__iter_traits**
- template<typename _Tp>
  using **std::__detail::__iter_value_t**
- template<typename _Tp>
  using **std::__detail::__range_iter_t**

- template<typename _Tp>
  using **std::__detail::__with_ref**
- template<typename _Fn, typename... _Is>
  using **std::indirect_result_t**
- template<indirectly_readable _Tp>
  using **std::iter_common_reference_t**
- template<typename _Tp>
  using **std::iter_difference_t**
- template<__detail::__dereferenceable _Tp>
  using **std::iter_reference_t**
- template<__detail::__dereferenceable _Tp>
  using std::iter_rvalue_reference_t
- template<typename _Tp>
  using **std::iter_value_t**
- template<indirectly_readable _Iter, indirectly_regular_unary_invocable< _Iter > _Proj>
  using std::projected
- template<indirectly_readable _Iter, indirectly_regular_unary_invocable< _Iter > _Proj>
  using **std::projected_value_t**

Functions

- template<typename _Tp>
  constexpr auto **std::ranges::__access::__begin** (_Tp &__t)
- void **std::ranges::__access::begin** ()=delete

Variables

- constexpr default_sentinel_t std::default_sentinel
- template<typename _Sent, typename _Iter>
  constexpr bool **std::disable_sized_sentinel_for**
- constexpr __imove::_IterMove **std::ranges::iter_move**
- constexpr __iswap::_IterSwap **std::ranges::iter_swap**
- constexpr unreachable_sentinel_t **std::unreachable_sentinel**

7.88.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

7.89 list.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX20_ONLY**(__expr)

7.89.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <list>.

7.90 locale_classes.h File Reference

Classes

- class std::__cxx11::collate< _CharT >
- class std::__cxx11::collate_byname< _CharT >
- class std::locale::facet
- class std::locale::id
- class std::locale

Namespaces

- namespace std

Variables

- template<typename _CharT>
  locale::id **std::collate< _CharT >::id**

7.90.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.91 locale_classes.tcc File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_STD_FACET**(...)

Functions

- template<typename _Facet>
  const _Facet * **std::__try_use_facet** (const locale &__loc)
- template<typename _Facet>
  bool std::has_facet (const locale &__loc)
- template<typename _Facet>
  const _Facet & std::use_facet (const locale &__loc)

7.91.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.92 locale_conv.h File Reference

Classes

- class std::wbuffer_convert< _Codecvt, _Elem, _Tr >
- class std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >

Namespaces

- namespace std

7.92.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.93 locale_facets.h File Reference

Classes

- class std::__ctype_abstract_base< _CharT >
- class std::ctype< _CharT >
- class std::ctype< char >
- class std::ctype< wchar_t >
- class std::ctype_byname< _CharT >
- class std::ctype_byname< char >
- class std::num_get< _CharT, _InIter >
- class std::num_put< _CharT, _OutIter >
- class std::numpunct< _CharT >
- class std::numpunct_byname< _CharT >

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_NUM_CXX11_FACETS**
- #define **_GLIBCXX_NUM_FACETS**
- #define **_GLIBCXX_NUM_LBDL_ALT128_FACETS**
- #define **_GLIBCXX_NUM_UNICODE_FACETS**

Functions

- template<typename _CharT>
  _CharT * **std::__add_grouping** (_CharT *__s, _CharT __sep, const char *__gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)
- template<typename _Tp>
  void **std::__convert_to_v** (const char *, _Tp &, ios_base::iostate &, const __c_locale &) throw ()
- template<> void **std::__convert_to_v** (const char *, double &, ios_base::iostate &, const __c_locale &) throw ()
- template<> void **std::__convert_to_v** (const char *, float &, ios_base::iostate &, const __c_locale &) throw ()
- template<> void **std::__convert_to_v** (const char *, long double &, ios_base::iostate &, const __c_locale &) throw ()
- template<typename _CharT, typename _OutIter>
  _OutIter **std::__write** (_OutIter __s, const _CharT *__ws, int __len)
- template<typename _CharT>
  ostreambuf_iterator< _CharT > **std::__write** (ostreambuf_iterator< _CharT > __s, const _CharT *__ws, int __len)
- template<typename _CharT>
  bool std::isalnum (_CharT __c, const locale &__loc)

- template<typename _CharT>
  bool std::isalpha (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isblank (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::iscntrl (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isdigit (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isgraph (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::islower (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isprint (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::ispunct (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isspace (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isupper (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isxdigit (_CharT __c, const locale &__loc)
- template<typename _CharT>
  _CharT std::tolower (_CharT __c, const locale &__loc)
- template<typename _CharT>
  _CharT std::toupper (_CharT __c, const locale &__loc)

Variables

- template<typename _CharT>
  locale::id **std::ctype< _CharT >::id**
- template<typename _CharT, typename _InIter>
  locale::id **std::num_get< _CharT, _InIter >::id**
- template<typename _CharT, typename _OutIter>
  locale::id **std::num_put< _CharT, _OutIter >::id**
- template<typename _CharT>
  locale::id **std::numpunct< _CharT >::id**

7.93.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.94 locale_facets.tcc File Reference

Namespaces

- namespace std

Functions

- template<typename _CharT>
  _CharT * **std::__add_grouping** (_CharT *__s, _CharT __sep, const char *__gbeg, size_t __gsize, const _CharT *__first, const _CharT *__last)

- template<typename _CharT, typename _ValueT>
  int **std::__int_to_char** (_CharT *__bufend, _ValueT __v, const _CharT *__lit, ios_base::fmtflags __flags, bool __dec)
- bool **std::__verify_grouping** (const char *__grouping, size_t __grouping_size, const string &__grouping_tmp) throw ()

7.94.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.95 locale_facets_nonio.h File Reference

Classes

- class std::messages< _CharT >
- struct std::messages_base
- class std::messages_byname< _CharT >
- class std::money_base
- class std::money_get< _CharT, _InIter >
- class std::money_put< _CharT, _OutIter >
- class std::moneypunct< _CharT, _Intl >
- class std::moneypunct_byname< _CharT, _Intl >
- class std::time_base
- class std::time_get< _CharT, _InIter >
- class std::time_get_byname< _CharT, _InIter >
- class std::time_put< _CharT, _OutIter >
- class std::time_put_byname< _CharT, _OutIter >

Namespaces

- namespace std

Variables

- template<typename _CharT>
  locale::id **std::__timepunct< _CharT >::id**
- template<typename _CharT>
  const _CharT * **std::__timepunct_cache< _CharT >::_S_timezones** [14]
- template<> const char * **std::__timepunct_cache< char >::_S_timezones** [14]
- template<> const wchar_t * **std::__timepunct_cache< wchar_t >::_S_timezones** [14]
- template<typename _CharT>
  locale::id **std::messages< _CharT >::id**
- template<typename _CharT, typename _InIter>
  locale::id **std::money_get< _CharT, _InIter >::id**
- template<typename _CharT, typename _OutIter>
  locale::id **std::money_put< _CharT, _OutIter >::id**
- template<typename _CharT, bool _Intl>
  locale::id **std::moneypunct< _CharT, _Intl >::id**
- template<typename _CharT, bool _Intl>
  const bool **std::moneypunct< _CharT, _Intl >::intl**
- template<typename _CharT, bool _Intl>
  const bool **std::moneypunct_byname< _CharT, _Intl >::intl**

- template<typename _CharT, typename _InIter>
  locale::id **std::time_get< _CharT, _InIter >::id**
- template<typename _CharT, typename _OutIter>
  locale::id **std::time_put< _CharT, _OutIter >::id**

7.95.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.96 locale_facets_nonio.tcc File Reference

Namespaces

- namespace std

7.96.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.97 localefwd.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Facet>
  bool std::has_facet (const locale &__loc)
- template<typename _CharT>
  bool std::isalnum (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isalpha (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isblank (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::iscntrl (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isdigit (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isgraph (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::islower (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isprint (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::ispunct (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isspace (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isupper (_CharT __c, const locale &__loc)
- template<typename _CharT>
  bool std::isxdigit (_CharT __c, const locale &__loc)

- template<typename _CharT>
  _CharT std::tolower (_CharT __c, const locale &__loc)
- template<typename _CharT>
  _CharT std::toupper (_CharT __c, const locale &__loc)
- template<typename _Facet>
  const _Facet & std::use_facet (const locale &__loc)

7.97.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.98 mask_array.h File Reference

Classes

- class std::mask_array< _Tp >

Namespaces

- namespace std

7.98.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

7.99 max_size_type.h File Reference

7.99.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

7.100 memory_resource.h File Reference

Classes

- struct std::allocator_traits< pmr::polymorphic_allocator< _Tp > >
- class std::pmr::memory_resource
- class std::pmr::polymorphic_allocator< _Tp >

Namespaces

- namespace std

Functions

- bool **std::pmr::operator==** (const memory_resource &__a, const memory_resource &__b) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::pmr::operator==** (const polymorphic_allocator< _Tp1 > &__a, const polymorphic_allocator< _Tp2 > &__b) noexcept

7.100.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory_resource>.

7.101 memoryfwd.h File Reference

Namespaces

- namespace std

7.101.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.102 mofunc_impl.h File Reference

Classes

- class std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_MOF_CV_REF**
- #define **_GLIBCXX_MOF_INV_QUALS**
- #define **_GLIBCXX_MOF_REF**

7.102.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

7.103 monostate.h File Reference

Namespaces

- namespace std

Functions

- constexpr strong_ordering **std::operator<=>** (monostate, monostate) noexcept
- constexpr bool **std::operator==** (monostate, monostate) noexcept

7.103.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.

7.104 move.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_FORWARD**(_Tp, __val)
- #define **_GLIBCXX_FWDREF**(_Tp)
- #define **_GLIBCXX_MOVE**(__val)

Typedefs

- template<typename _Tp, typename _Up>
  using **std::__like_t**

Functions

- template<typename _Tp>
  constexpr _Tp * std::__addressof (_Tp &__r) noexcept
- template<typename _Tp, typename _Up = _Tp>
  constexpr _Tp **std::__exchange** (_Tp &__obj, _Up &&__new_val)
- template<typename _Tp>
  constexpr _Tp * std::addressof (_Tp &__r) noexcept
- template<typename _Tp>
  const _Tp * **std::addressof** (const _Tp &&)=delete
- template<typename _Tp>
  constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &&__t) noexcept
- template<typename _Tp>
  constexpr _Tp && std::forward (typename std::remove_reference< _Tp >::type &__t) noexcept
- template<typename _Tp, typename _Up>
  constexpr __like_t< _Tp, _Up > std::forward_like (_Up &&__x) noexcept
- template<typename _Tp>
  constexpr std::remove_reference< _Tp >::type && std::move (_Tp &&__t) noexcept
- template<typename _Tp>
  constexpr __conditional_t< __move_if_noexcept_cond< _Tp >::value, const _Tp &, _Tp && > std::move_if_noexcept (_Tp &__x) noexcept
- template<typename _Tp>
  constexpr void std::swap (_Tp &__a, _Tp &__b) noexcept(/*conditional */) is_nothrow_move_assignable< _Tp > >
- template<typename _Tp, size_t _Nm>
  constexpr void std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]) noexcept(/*conditional */)

7.104.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.

7.105 nested_exception.h File Reference

Classes

- class std::nested_exception

Namespaces

- namespace std

Functions

- template<typename _Ex>
  void std::rethrow_if_nested (const _Ex &__ex)
- template<typename _Tp>
  void std::throw_with_nested (_Tp &&__t)

7.105.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <exception>.

7.106 bits/new_allocator.h File Reference

Classes

- class std::__new_allocator< _Tp >

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_OPERATOR_DELETE**
- #define **_GLIBCXX_OPERATOR_NEW**
- #define **_GLIBCXX_SIZED_DEALLOC**(p, n)

7.106.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.107 ext/new_allocator.h File Reference

Classes

- class __gnu_cxx::new_allocator< _Tp >

Namespaces

- namespace __gnu_cxx

7.107.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.108 new_except.h File Reference

Classes

- class std::bad_alloc

Namespaces

- namespace std

7.108.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <new>.

7.109 new_throw.h File Reference

Namespaces

- namespace std

Functions

- void **std::__throw_bad_alloc** (void)
- void **std::__throw_bad_array_new_length** (void)

7.109.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <new>.

7.110 node_handle.h File Reference

7.110.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <map,set,unordered_map,unordered_set>.

7.111 optional_ref.h File Reference

7.111.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <optional>.

7.112 ostream.h File Reference

Classes

- class std::basic_ostream< _CharT, _Traits >
- class std::basic_ostream< _CharT, _Traits >::sentry

Namespaces

- namespace std

Typedefs

- template<typename _Os, typename _Tp>
  using **std::__rvalue_stream_insertion_t**

Functions

- template<typename _Ostream, typename _Tp>
  __rvalue_stream_insertion_t< _Ostream, _Tp > std::operator<< (_Ostream &&__os, const _Tp &__x)
- template<typename _Traits>
  basic_ostream< char, _Traits > & **std::operator**<< (basic_ostream< char, _Traits > &, char16_t)=delete

- template<typename _Traits>
  basic_ostream< char, _Traits > & **std::operator<<** (basic_ostream< char, _Traits > &, char32_t)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & **std::operator<<** (basic_ostream< char, _Traits > &, const char16_t *)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & **std::operator<<** (basic_ostream< char, _Traits > &, const char32_t *)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & **std::operator<<** (basic_ostream< char, _Traits > &, const wchar_t *)=delete
- template<typename _Traits>
  basic_ostream< char, _Traits > & **std::operator<<** (basic_ostream< char, _Traits > &, wchar_t)=delete
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & **std::operator<<** (basic_ostream< wchar_t, _Traits > &, char16_t)=delete
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & **std::operator<<** (basic_ostream< wchar_t, _Traits > &, char32_t)=delete
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & **std::operator<<** (basic_ostream< wchar_t, _Traits > &, const char16_t *)=delete
- template<typename _Traits>
  basic_ostream< wchar_t, _Traits > & **std::operator<<** (basic_ostream< wchar_t, _Traits > &, const char32_t *)=delete

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, _CharT __c)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, char __c)
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, char __c)
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, signed char __c)
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, unsigned char __c)

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const _CharT *__s)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const char *__s)
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const signed char *__s)
- template<typename _Traits>
  basic_ostream< char, _Traits > & std::operator<< (basic_ostream< char, _Traits > &__out, const unsigned char *__s)

7.112.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ostream>.

7.113 ostream.tcc File Reference

Namespaces

- namespace std

Functions

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__out, const char *__s)

7.113.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ostream>.

7.114 ostream_insert.h File Reference

Namespaces

- namespace std

7.114.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ostream>.

7.115 ostream_print.h File Reference

7.115.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ostream>.
This file contains the parts of <ostream> which are currently defined inline, but should be moved into the library eventually.

7.116 out_ptr.h File Reference

Classes

- class std::inout_ptr_t< _Smart, _Pointer, _Args >
- class std::out_ptr_t< _Smart, _Pointer, _Args >

Namespaces

- namespace std

Functions

- template<typename _Pointer = void, typename _Smart, typename... _Args>
  auto std::inout_ptr (_Smart &__s, _Args &&... __args)
- template<typename _Pointer = void, typename _Smart, typename... _Args>
  auto std::out_ptr (_Smart &__s, _Args &&... __args)

7.116.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.117 parse_numbers.h File Reference

Namespaces

- namespace std

Typedefs

- template<unsigned long long _Val>
  using **std::__parse_int::__ull_constant**
- template<char... _Digs>
  using **std::__select_int::_Select_int**

7.117.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <chrono>.

7.118 postypes.h File Reference

Classes

- class std::fpos< _StateT >

Namespaces

- namespace std

Typedefs

- typedef long long std::streamoff
- typedef fpos< mbstate_t > std::streampos
- typedef ptrdiff_t std::streamsize
- typedef fpos< mbstate_t > std::u16streampos
- typedef fpos< mbstate_t > std::u32streampos
- typedef fpos< mbstate_t > std::wstreampos

Functions

- template<typename _StateT>
  bool **std::operator!=** (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)
- template<typename _StateT>
  bool std::operator== (const fpos< _StateT > &__lhs, const fpos< _StateT > &__rhs)

7.118.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

7.119 predefined_ops.h File Reference

Namespaces

- namespace __gnu_cxx

Typedefs

- template<typename _Fn>
  using **__gnu_cxx::__ops::__by_ref_or_value_fn**
- template<typename _Func, typename _Value>
  using **__gnu_cxx::__ops::_Comp_with_val_1st**
- template<typename _Func, typename _Value>
  using **__gnu_cxx::__ops::_Comp_with_val_2nd**
- typedef std::equal_to< void > **__gnu_cxx::__ops::equal_to**
- typedef std::less< void > **__gnu_cxx::__ops::less**

Functions

- template<typename _Value>
  constexpr _Comp_with_val_2nd< equal_to, _Value > **__gnu_cxx::__ops::__equal_to** (const _Value &__val)
- template<typename _Func, typename _Value>
  constexpr _Comp_with_val_1st< _Func, _Value > **__gnu_cxx::__ops::bind1st** (_Func &__f, const _Value &↩ __val)
- template<typename _Func, typename _Value>
  constexpr _Comp_with_val_2nd< _Func, _Value > **__gnu_cxx::__ops::bind2nd** (_Func &__f, const _Value &__val)
- template<typename _Func>
  constexpr _Unary_negate< _Func > **__gnu_cxx::__ops::not1** (_Func &__f)

7.119.1 Detailed Description

This is an internal header file, included by other library headers. You should not attempt to use it directly. Instead, include <algorithm>.

7.120 print.h File Reference

7.120.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <print>.
This file contains the parts of <print> which are currently defined inline, but should be moved into the library eventually.

7.121 ptr_traits.h File Reference

Classes

- struct std::pointer_traits< _Ptr >
- struct std::pointer_traits< _Tp * >

Namespaces

- namespace __gnu_debug
- namespace std

Typedefs

- template<typename _Ptr, typename _Tp>
  using std::__ptr_rebind

Functions

- template<typename _Tp>
  constexpr _Tp * std::to_address (_Tp *__ptr) noexcept
- template<typename _Ptr>
  constexpr auto std::to_address (const _Ptr &__ptr) noexcept

7.121.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.122 quoted_string.h File Reference

Classes

- struct std::__detail::_Quoted_string< _String, _CharT >

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _CharT, typename _Traits, typename _String>
  std::basic_ostream< _CharT, _Traits > & std::__detail::operator<< (std::basic_ostream< _CharT, _Traits > &↩
  __os, const _Quoted_string< _String, _CharT > &__str)
- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::__detail::operator<< (std::basic_ostream< _CharT, _Traits > &↩
  __os, const _Quoted_string< const _CharT *, _CharT > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc>
  std::basic_istream< _CharT, _Traits > & std::__detail::operator>> (std::basic_istream< _CharT, _Traits > &_↩
  _is, const _Quoted_string< basic_string< _CharT, _Traits, _Alloc > &, _CharT > &__str)

7.122.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iomanip>.

7.123 random.h File Reference

Classes

- class std::bernoulli_distribution
- class std::binomial_distribution< _IntType >
- class std::cauchy_distribution< _RealType >
- class std::chi_squared_distribution< _RealType >
- class std::discard_block_engine< _RandomNumberEngine, __p, __r >
- class std::discrete_distribution< _IntType >

- class std::exponential_distribution< _RealType >
- class std::extreme_value_distribution< _RealType >
- class std::fisher_f_distribution< _RealType >
- class std::gamma_distribution< _RealType >
- class std::geometric_distribution< _IntType >
- class std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >
- class std::linear_congruential_engine< _UIntType, __a, __c, __m >
- class std::lognormal_distribution< _RealType >
- class std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >
- class std::negative_binomial_distribution< _IntType >
- class std::normal_distribution< _RealType >
- struct std::uniform_real_distribution< _RealType >::param_type
- struct std::normal_distribution< _RealType >::param_type
- struct std::lognormal_distribution< _RealType >::param_type
- struct std::gamma_distribution< _RealType >::param_type
- struct std::chi_squared_distribution< _RealType >::param_type
- struct std::cauchy_distribution< _RealType >::param_type
- struct std::fisher_f_distribution< _RealType >::param_type
- struct std::student_t_distribution< _RealType >::param_type
- struct std::bernoulli_distribution::param_type
- struct std::binomial_distribution< _IntType >::param_type
- struct std::geometric_distribution< _IntType >::param_type
- struct std::negative_binomial_distribution< _IntType >::param_type
- struct std::poisson_distribution< _IntType >::param_type
- struct std::exponential_distribution< _RealType >::param_type
- struct std::weibull_distribution< _RealType >::param_type
- struct std::extreme_value_distribution< _RealType >::param_type
- struct std::discrete_distribution< _IntType >::param_type
- struct std::piecewise_constant_distribution< _RealType >::param_type
- struct std::piecewise_linear_distribution< _RealType >::param_type
- class std::philox_engine< _UIntType, __w, __n, __r, __consts >
- class std::piecewise_constant_distribution< _RealType >
- class std::piecewise_linear_distribution< _RealType >
- class std::poisson_distribution< _IntType >
- class std::random_device
- class std::seed_seq
- class std::shuffle_order_engine< _RandomNumberEngine, __k >
- class std::student_t_distribution< _RealType >
- class std::subtract_with_carry_engine< _UIntType, __w, __s, __r >
- class std::uniform_real_distribution< _RealType >
- class std::weibull_distribution< _RealType >

Namespaces

- namespace std
- namespace std::__detail

Typedefs

- template<typename _Tp>
  using **std::__detail::__piecewise_distributions_storage_t**
- typedef minstd_rand0 **std::default_random_engine**
- typedef shuffle_order_engine< minstd_rand0, 256 > **std::knuth_b**
- typedef linear_congruential_engine< uint_fast32_t, 48271UL, 0UL, 2147483647UL > std::minstd_rand
- typedef linear_congruential_engine< uint_fast32_t, 16807UL, 0UL, 2147483647UL > std::minstd_rand0
- typedef mersenne_twister_engine< uint_fast32_t, 32, 624, 397, 31, 0x9908b0dfUL, 11, 0xffffffffUL, 7, 0x9d2c5680UL, 15, 0xefc60000UL, 18, 1812433253UL > std::mt19937
- typedef mersenne_twister_engine< uint_fast64_t, 64, 312, 156, 31, 0xb5026f5aa96619e9ULL, 29, 0x5555555555555555ULL, 17, 0x71d67fffeda60000ULL, 37, 0xfff7eee000000000ULL, 43, 6364136223846793005ULL > std::mt19937_64
- typedef philox_engine< uint_fast32_t, 32, 4, 10, 0xCD9E8D57, 0x9E3779B9, 0xD2511F53, 0xBB67AE85 > std::philox4x32
- typedef philox_engine< uint_fast64_t, 64, 4, 10, 0xCA5A826395121157, 0x9E3779B97F4A7C15, 0xD2E7470↩EE14C6C93, 0xBB67AE8584CAA73B > std::philox4x64
- typedef discard_block_engine< ranlux24_base, 223, 23 > **std::ranlux24**
- typedef subtract_with_carry_engine< uint_fast32_t, 24, 10, 24 > **std::ranlux24_base**
- typedef discard_block_engine< ranlux48_base, 389, 11 > **std::ranlux48**
- typedef subtract_with_carry_engine< uint_fast64_t, 48, 5, 12 > **std::ranlux48_base**

Functions

- template<typename _RealType, size_t __bits, typename _UniformRandomNumberGenerator>
  _RealType std::generate_canonical (_UniformRandomNumberGenerator &__g)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)
- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)
- template<typename _RandomNumberEngine, size_t __w, typename _UIntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)

- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)
- template<typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::bernoulli_distribution &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::extreme_value_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::geometric_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, std::weibull_distribution< _RealType > &__x)

7.123.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <random>.

7.124 bits/random.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_GEN_CANON_CONST**

Functions

- template<typename _ValT, typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::__detail::__extract_params** (basic_istream< _CharT, _Traits > &__is, vector< _ValT > &__vals, size_t __n)
- template<typename _UInt1, typename _UInt2, typename _UComm = __conditional_t<(sizeof(_UInt2) > sizeof(_UInt1)), _UInt2, _UInt1>>
  constexpr __gen_canon_log_res< _UInt1 > **std::__gen_canon_log** (_UInt1 __val, _UInt2 __base)
- template<typename _RealT, size_t __d, typename _Urbg>
  _RealT **std::__generate_canonical_any** (_Urbg &__urng)
- template<typename _RealT, size_t __d, typename _Urbg>
  _RealT **std::__generate_canonical_pow2** (_Urbg &__urng)
- template<typename _Rng>
  constexpr bool **std::__is_power_of_2_less_1** (_Rng __range)

- template<typename _InputIterator, typename _RealType>
  bool **std::__detail::__load_first2** (_InputIterator &__first, _InputIterator __last, _RealType *__out)
- template<typename _Tp>
  constexpr bool **std::__detail::__p1_representable_as_double** (_Tp __x) noexcept
- template<typename _Tp>
  constexpr bool **std::__detail::__representable_as_double** (_Tp __x) noexcept
- template<typename _RealT, size_t __digits, typename _Urbg>
  _RealT std::generate_canonical (_Urbg &__urng)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &, const std::uniform_real_distribution< _RealType > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const chi_squared_distribution< _RealType > &__x)
- template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const discard_block_engine< _RandomNumberEngine, __p, __r > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const discrete_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const fisher_f_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const gamma_distribution< _RealType > &__x)
- template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const lognormal_distribution< _RealType > &__x)
- template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _↩
  UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const negative_binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const normal_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_constant_distribution< _RealType > &__x)

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const piecewise_linear_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const poisson_distribution< _IntType > &__x)
- template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const shuffle_order_engine< _RandomNumberEngine, __k > &__x)
- template<typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::bernoulli_distribution &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::extreme_value_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::geometric_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const std::weibull_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const student_t_distribution< _RealType > &__x)
- template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **std::operator**<< (std::basic_ostream< _CharT, _Traits > &__os, const subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)
- template<typename _RealType>
  bool **std::operator==** (const std::normal_distribution< _RealType > &__d1, const std::normal_distribution< _RealType > &__d2)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_int_distribution< _IntType > &)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &, std::uniform_real_distribution< _RealType > &)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator**>> (std::basic_istream< _CharT, _Traits > &__is, binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator**>> (std::basic_istream< _CharT, _Traits > &__is, chi_squared_distribution< _RealType > &__x)
- template<typename _RandomNumberEngine, size_t __p, size_t __r, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator**>> (std::basic_istream< _CharT, _Traits > &__is, discard_block_engine< _RandomNumberEngine, __p, __r > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator**>> (std::basic_istream< _CharT, _Traits > &__is, discrete_distribution< _IntType > &__x)

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, fisher_f_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, gamma_distribution< _RealType > &__x)
- template<typename _UIntType, _UIntType __a, _UIntType __c, _UIntType __m, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, linear_congruential_engine< _UIntType, __a, __c, __m > &__lcr)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, lognormal_distribution< _RealType > &__x)
- template<typename _UIntType, size_t __w, size_t __n, size_t __m, size_t __r, _UIntType __a, size_t __u, _UIntType __d, size_t __s, _↩
  UIntType __b, size_t __t, _UIntType __c, size_t __l, _UIntType __f, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >
  &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, negative_binomial_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, normal_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, piecewise_constant_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, piecewise_linear_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, poisson_distribution< _IntType > &__x)
- template<typename _RandomNumberEngine, size_t __k, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, shuffle_order_engine< _RandomNumberEngine, __k > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,
  std::cauchy_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,
  std::exponential_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,
  std::extreme_value_distribution< _RealType > &__x)
- template<typename _IntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,
  std::geometric_distribution< _IntType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is,
  std::weibull_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__↩
  is, student_t_distribution< _RealType > &__x)

- template<typename _UIntType, size_t __w, size_t __s, size_t __r, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **std::operator>>** (std::basic_istream< _CharT, _Traits > &__is, subtract_with_carry_engine< _UIntType, __w, __s, __r > &__x)

Variables

- template<typename _Tp>
  const bool **std::__is_rand_dist_float_v**

7.124.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <random>.

7.125 ext/random.tcc File Reference

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::beta_distribution< _RealType > &__x)
- template<typename _UIntType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)
- template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)
- template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::triangular_distribution< _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::von_mises_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const arcsine_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const hoyt_distribution< _RealType > &__x)

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const k_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const logistic_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const nakagami_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const pareto_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩
  __os, const rice_distribution< _RealType > &__x)
- template<size_t _Dimen, typename _RealType>
  bool **__gnu_cxx::operator==** (const __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d1, const
  __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__d2)
- template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t
  __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4>
  bool **__gnu_cxx::operator==** (const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __↩
  pos1, __sl1, __sl2, __sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4
  > &__lhs, const __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __↩
  sr1, __sr2, __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__rhs)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::beta_distribution< _RealType > &__x)
- template<typename _UIntType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::hypergeometric_distribution< _UIntType > &__x)
- template<size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::normal_mv_distribution< _Dimen, _RealType > &__x)
- template<typename _UIntType, size_t __m, size_t __pos1, size_t __sl1, size_t __sl2, size_t __sr1, size_t __sr2, uint32_t __msk1, uint32_t
  __msk2, uint32_t __msk3, uint32_t __msk4, uint32_t __parity1, uint32_t __parity2, uint32_t __parity3, uint32_t __parity4, typename _CharT,
  typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::simd_fast_mersenne_twister_engine< _UIntType, __m, __pos1, __sl1, __sl2, __sr1, __sr2,
  __msk1, __msk2, __msk3, __msk4, __parity1, __parity2, __parity3, __parity4 > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::triangular_distribution< _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::uniform_inside_sphere_distribution< _Dimen, _RealType > &__x)
- template<std::size_t _Dimen, typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::uniform_on_sphere_distribution< _Dimen, _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, __gnu_cxx::von_mises_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, arcsine_distribution< _RealType > &__x)

- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, hoyt_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, k_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, logistic_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, nakagami_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, pareto_distribution< _RealType > &__x)
- template<typename _RealType, typename _CharT, typename _Traits>
  std::basic_istream< _CharT, _Traits > & **__gnu_cxx::operator>>** (std::basic_istream< _CharT, _Traits > &↩
  __is, rice_distribution< _RealType > &__x)

7.125.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/random>.

7.126 range_access.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Container>
  constexpr auto std::begin (_Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.begin())
- template<typename _Tp, size_t _Nm>
  constexpr _Tp * std::begin (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto std::begin (const _Container &__cont) noexcept(noexcept(__cont.begin())) -> decltype(__cont.↩
  begin())
- template<typename _Container>
  constexpr auto std::cbegin (const _Container &__cont) noexcept(noexcept(std::begin(__cont))) -> decltype(std::begin(↩
  __cont))
- template<typename _Container>
  constexpr auto std::cend (const _Container &__cont) noexcept(noexcept(std::end(__cont))) -> decltype(std::end(↩
  __cont))
- template<typename _Container>
  constexpr auto std::crbegin (const _Container &__cont) noexcept(noexcept(std::rbegin(__cont))) -> decltype(std::rbegin(↩
  __cont))
- template<typename _Container>
  constexpr auto std::crend (const _Container &__cont) noexcept(noexcept(std::rend(__cont))) -> decltype(std::rend(↩
  __cont))
- template<typename _Container>
  constexpr auto std::data (_Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.data())

- template<typename _Tp, size_t _Nm>
  constexpr _Tp * std::data (_Tp(&__array)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto std::data (const _Container &__cont) noexcept(noexcept(__cont.data())) -> decltype(__cont.↩ data())
- template<typename _Container>
  constexpr auto std::empty (const _Container &__cont) noexcept(noexcept(__cont.empty())) -> decltype(__↩ cont.empty())
- template<typename _Tp, size_t _Nm>
  constexpr bool std::empty (const _Tp(&)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto std::end (_Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())
- template<typename _Tp, size_t _Nm>
  constexpr _Tp * std::end (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto std::end (const _Container &__cont) noexcept(noexcept(__cont.end())) -> decltype(__cont.end())
- template<typename _Container>
  constexpr auto std::rbegin (_Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__cont.↩ rbegin())
- template<typename _Tp, size_t _Nm>
  constexpr reverse_iterator< _Tp * > std::rbegin (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto std::rbegin (const _Container &__cont) noexcept(noexcept(__cont.rbegin())) -> decltype(__↩ cont.rbegin())
- template<typename _Tp>
  constexpr reverse_iterator< const _Tp * > std::rbegin (initializer_list< _Tp > __il) noexcept
- template<typename _Container>
  constexpr auto std::rend (_Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.rend())
- template<typename _Tp, size_t _Nm>
  constexpr reverse_iterator< _Tp * > std::rend (_Tp(&__arr)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto std::rend (const _Container &__cont) noexcept(noexcept(__cont.rend())) -> decltype(__cont.↩ rend())
- template<typename _Tp>
  constexpr reverse_iterator< const _Tp * > std::rend (initializer_list< _Tp > __il) noexcept
- template<typename _Container>
  constexpr auto std::size (const _Container &__cont) noexcept(noexcept(__cont.size())) -> decltype(__cont.size())
- template<typename _Tp, size_t _Nm>
  constexpr size_t std::size (const _Tp(&)[_Nm]) noexcept
- template<typename _Container>
  constexpr auto std::ssize (const _Container &__cont) noexcept(noexcept(__cont.size())) -> common_type_t< ptrdiff_t, make_signed_t< decltype(__cont.size())> >
- template<typename _Tp, ptrdiff_t _Num>
  constexpr ptrdiff_t std::ssize (const _Tp(&)[_Num]) noexcept

7.126.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

7.127 ranges_algo.h File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Fp>
  using **std::ranges::__detail::__by_ref_or_value_fn**
- template<typename _Iter1, typename _Iter2, typename _Out>
  using **std::ranges::binary_transform_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::copy_if_result**
- template<typename _Iter, typename _Tp>
  using **std::ranges::fold_left_first_with_iter_result**
- template<typename _Iter, typename _Tp>
  using **std::ranges::fold_left_with_iter_result**
- template<typename _Iter, typename _Fp>
  using **std::ranges::for_each_n_result**
- template<typename _Iter, typename _Fp>
  using **std::ranges::for_each_result**
- template<typename _Iter1, typename _Iter2, typename _Out>
  using **std::ranges::merge_result**
- template<typename _Iter>
  using **std::ranges::minmax_element_result**
- template<typename _Tp>
  using **std::ranges::minmax_result**
- template<typename _Iter>
  using **std::ranges::next_permutation_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::partial_sort_copy_result**
- template<typename _Iter, typename _Out1, typename _Out2>
  using **std::ranges::partition_copy_result**
- template<typename _Iter>
  using **std::ranges::prev_permutation_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::remove_copy_if_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::remove_copy_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::replace_copy_if_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::replace_copy_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::reverse_copy_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::rotate_copy_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::set_difference_result**
- template<typename _Iter1, typename _Iter2, typename _Out>
  using **std::ranges::set_intersection_result**
- template<typename _Iter1, typename _Iter2, typename _Out>
  using **std::ranges::set_symmetric_difference_result**
- template<typename _Iter1, typename _Iter2, typename _Out>
  using **std::ranges::set_union_result**
- template<typename _Iter1, typename _Iter2>
  using **std::ranges::swap_ranges_result**

- template<typename _Iter, typename _Out>
  using **std::ranges::unary_transform_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::unique_copy_result**

Functions

- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__adjust_heap** (_Iter __first, iter_difference_t< _Iter > __holeIndex, iter_difference_t< _Iter > __len, iter_value_t< _Iter > __value, _Comp __comp)
- template<typename _Iter, typename _Distance, typename _Compare>
  constexpr void **std::ranges::__detail::__chunk_insertion_sort** (_Iter __first, _Iter __last, _Distance __chunk↩
  _size, _Compare __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__final_insertion_sort** (_Iter __first, _Iter __last, _Comp __comp)
- template<typename _Iter, typename _Pred, typename _Distance>
  constexpr _Iter **std::ranges::__detail::__find_if_not_n** (_Iter __first, _Distance &__len, _Pred __pred)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__heap_select** (_Iter __first, _Iter __middle, _Iter __last, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__inplace_stable_sort** (_Iter __first, _Iter __last, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__insertion_sort** (_Iter __first, _Iter __last, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__introselect** (_Iter __first, _Iter __nth, _Iter __last, iter_difference_t< _Iter > __depth_limit, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__introsort_loop** (_Iter __first, _Iter __last, unsigned __depth_limit, _↩
  Comp __comp)
- template<typename _Comp, typename _Proj>
  constexpr _Comp_proj< _Comp, _Proj > **std::ranges::__detail::__make_comp_proj** (_Comp &__comp, _Proj &__proj)
- template<typename _Pred, typename _Proj>
  constexpr _Pred_proj< _Pred, _Proj > **std::ranges::__detail::__make_pred_proj** (_Pred &__pred, _Proj &_↩
  _proj)
- template<typename _Iter, typename _Pointer, typename _Comp>
  void **std::ranges::__detail::__merge_adaptive** (_Iter __first, _Iter __middle, _Iter __last, iter_difference_t< _Iter > __len1, iter_difference_t< _Iter > __len2, _Pointer __buffer, _Comp __comp)
- template<typename _Iter, typename _Distance, typename _Pointer, typename _Comp>
  void **std::ranges::__detail::__merge_adaptive_resize** (_Iter __first, _Iter __middle, _Iter __last, _Distance _↩
  _len1, _Distance __len2, _Pointer __buffer, _Distance __buffer_size, _Comp __comp)
- template<typename _Iter, typename _Out, typename _Distance, typename _Comp>
  void **std::ranges::__detail::__merge_sort_loop** (_Iter __first, _Iter __last, _Out __result, _Distance __step_↩
  size, _Comp __comp)
- template<typename _Iter, typename _Pointer, typename _Comp>
  void **std::ranges::__detail::__merge_sort_with_buffer** (_Iter __first, _Iter __last, _Pointer __buffer, _Comp ↩
  __comp)
- template<typename _Iter, typename _Distance, typename _Comp>
  constexpr void **std::ranges::__detail::__merge_without_buffer** (_Iter __first, _Iter __middle, _Iter __last, _↩
  Distance __len1, _Distance __len2, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__move_median_to_first** (_Iter __result, _Iter __a, _Iter __b, _Iter __c, _Comp __comp)

- template<typename _Iter, typename _Out, typename _Comp>
  _Out **std::ranges::__detail::__move_merge** (_Iter __first1, _Iter __last1, _Iter __first2, _Iter __last2, _Out __result, _Comp __comp)
- template<typename _Iter1, typename _Iter2, typename _Out, typename _Comp>
  void **std::ranges::__detail::__move_merge_adaptive** (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Iter2 __last2, _Out __result, _Comp __comp)
- template<typename _Iter1, typename _Iter2, typename _Iter3, typename _Comp>
  void **std::ranges::__detail::__move_merge_adaptive_backward** (_Iter1 __first1, _Iter1 __last1, _Iter2 __first2, _Iter2 __last2, _Iter3 __result, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__partial_sort** (_Iter __first, _Iter __middle, _Iter __last, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__pop_heap** (_Iter __first, _Iter __last, _Iter __result, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__push_heap** (_Iter __first, iter_difference_t< _Iter > __holeIndex, iter_difference_t< _Iter > __topIndex, iter_value_t< _Iter > __value, _Comp __comp)
- template<typename _Iter1, typename _Iter2>
  _Iter1 **std::ranges::__detail::__rotate_adaptive** (_Iter1 __first, _Iter1 __middle, _Iter1 __last, iter_difference_t< _Iter1 > __len1, iter_difference_t< _Iter1 > __len2, _Iter2 __buffer, iter_difference_t< _Iter1 > __buffer_size)
- template<typename _Iter, typename _Sent, typename _Pointer, typename _Pred, typename _Distance>
  constexpr subrange< _Iter > **std::ranges::__detail::__stable_partition_adaptive** (_Iter __first, _Sent __last, _Pred __pred, _Distance __len, _Pointer __buffer, _Distance __buffer_size)
- template<typename _Iter, typename _Pointer, typename _Comp>
  void **std::ranges::__detail::__stable_sort_adaptive** (_Iter __first, _Iter __middle, _Iter __last, _Pointer __buffer, _Comp __comp)
- template<typename _Iter, typename _Pointer, typename _Distance, typename _Comp>
  void **std::ranges::__detail::__stable_sort_adaptive_resize** (_Iter __first, _Iter __last, _Pointer __buffer, _Distance __buffer_size, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__unguarded_insertion_sort** (_Iter __first, _Iter __last, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr void **std::ranges::__detail::__unguarded_linear_insert** (_Iter __last, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr _Iter **std::ranges::__detail::__unguarded_partition** (_Iter __first, _Iter __last, _Iter __pivot, _Comp __comp)
- template<typename _Iter, typename _Comp>
  constexpr _Iter **std::ranges::__detail::__unguarded_partition_pivot** (_Iter __first, _Iter __last, _Comp __comp)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator **std::shift_left** (_ForwardIterator __first, _ForwardIterator __last, typename iterator_traits< _ForwardIterator >::difference_type __n)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator **std::shift_right** (_ForwardIterator __first, _ForwardIterator __last, typename iterator_traits< _ForwardIterator >::difference_type __n)

Variables

- constexpr int **std::ranges::__detail::__sort_threshold**
- constexpr __all_of_fn **std::ranges::all_of**
- constexpr __any_of_fn **std::ranges::any_of**
- constexpr __binary_search_fn **std::ranges::binary_search**
- constexpr __clamp_fn **std::ranges::clamp**
- constexpr __contains_fn **std::ranges::contains**

- constexpr __contains_subrange_fn **std::ranges::contains_subrange**
- constexpr __copy_if_fn **std::ranges::copy_if**
- constexpr __count_fn **std::ranges::count**
- constexpr __count_if_fn **std::ranges::count_if**
- constexpr __ends_with_fn **std::ranges::ends_with**
- constexpr __equal_range_fn **std::ranges::equal_range**
- constexpr __find_end_fn **std::ranges::find_end**
- constexpr __find_first_of_fn **std::ranges::find_first_of**
- constexpr __find_last_fn **std::ranges::find_last**
- constexpr __find_last_if_fn **std::ranges::find_last_if**
- constexpr __find_last_if_not_fn **std::ranges::find_last_if_not**
- constexpr __fold_left_fn **std::ranges::fold_left**
- constexpr __fold_left_first_fn **std::ranges::fold_left_first**
- constexpr __fold_left_first_with_iter_fn **std::ranges::fold_left_first_with_iter**
- constexpr __fold_left_with_iter_fn **std::ranges::fold_left_with_iter**
- constexpr __fold_right_fn **std::ranges::fold_right**
- constexpr __fold_right_last_fn **std::ranges::fold_right_last**
- constexpr __for_each_fn **std::ranges::for_each**
- constexpr __for_each_n_fn **std::ranges::for_each_n**
- constexpr __generate_fn **std::ranges::generate**
- constexpr __generate_n_fn **std::ranges::generate_n**
- constexpr __includes_fn **std::ranges::includes**
- constexpr __inplace_merge_fn **std::ranges::inplace_merge**
- constexpr __is_heap_fn **std::ranges::is_heap**
- constexpr __is_heap_until_fn **std::ranges::is_heap_until**
- constexpr __is_partitioned_fn **std::ranges::is_partitioned**
- constexpr __is_permutation_fn **std::ranges::is_permutation**
- constexpr __is_sorted_fn **std::ranges::is_sorted**
- constexpr __is_sorted_until_fn **std::ranges::is_sorted_until**
- constexpr __lexicographical_compare_fn **std::ranges::lexicographical_compare**
- constexpr __lower_bound_fn **std::ranges::lower_bound**
- constexpr __make_heap_fn **std::ranges::make_heap**
- constexpr __max_fn **std::ranges::max**
- constexpr __max_element_fn **std::ranges::max_element**
- constexpr __merge_fn **std::ranges::merge**
- constexpr __min_element_fn **std::ranges::min_element**
- constexpr __minmax_fn **std::ranges::minmax**
- constexpr __minmax_element_fn **std::ranges::minmax_element**
- constexpr __next_permutation_fn **std::ranges::next_permutation**
- constexpr __none_of_fn **std::ranges::none_of**
- constexpr __nth_element_fn **std::ranges::nth_element**
- constexpr __partial_sort_fn **std::ranges::partial_sort**
- constexpr __partial_sort_copy_fn **std::ranges::partial_sort_copy**
- constexpr __partition_fn **std::ranges::partition**
- constexpr __partition_copy_fn **std::ranges::partition_copy**
- constexpr __partition_point_fn **std::ranges::partition_point**
- constexpr __pop_heap_fn **std::ranges::pop_heap**
- constexpr __prev_permutation_fn **std::ranges::prev_permutation**
- constexpr __push_heap_fn **std::ranges::push_heap**
- constexpr __remove_fn **std::ranges::remove**
- constexpr __remove_copy_fn **std::ranges::remove_copy**

- constexpr __remove_copy_if_fn **std::ranges::remove_copy_if**
- constexpr __remove_if_fn **std::ranges::remove_if**
- constexpr __replace_fn **std::ranges::replace**
- constexpr __replace_copy_fn **std::ranges::replace_copy**
- constexpr __replace_copy_if_fn **std::ranges::replace_copy_if**
- constexpr __replace_if_fn **std::ranges::replace_if**
- constexpr __reverse_fn **std::ranges::reverse**
- constexpr __reverse_copy_fn **std::ranges::reverse_copy**
- constexpr __rotate_fn **std::ranges::rotate**
- constexpr __rotate_copy_fn **std::ranges::rotate_copy**
- constexpr __sample_fn **std::ranges::sample**
- constexpr __search_n_fn **std::ranges::search_n**
- constexpr __set_difference_fn **std::ranges::set_difference**
- constexpr __set_intersection_fn **std::ranges::set_intersection**
- constexpr __set_symmetric_difference_fn **std::ranges::set_symmetric_difference**
- constexpr __set_union_fn **std::ranges::set_union**
- constexpr __shift_left_fn **std::ranges::shift_left**
- constexpr __shift_right_fn **std::ranges::shift_right**
- constexpr __shuffle_fn **std::ranges::shuffle**
- constexpr __sort_fn **std::ranges::sort**
- constexpr __sort_heap_fn **std::ranges::sort_heap**
- constexpr __stable_partition_fn **std::ranges::stable_partition**
- constexpr __stable_sort_fn **std::ranges::stable_sort**
- constexpr __starts_with_fn **std::ranges::starts_with**
- constexpr __swap_ranges_fn **std::ranges::swap_ranges**
- constexpr __transform_fn **std::ranges::transform**
- constexpr __unique_fn **std::ranges::unique**
- constexpr __unique_copy_fn **std::ranges::unique_copy**
- constexpr __upper_bound_fn **std::ranges::upper_bound**

7.127.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <algorithm>.

7.128 ranges_algobase.h File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Iter1, typename _Iter2>
  using **std::ranges::copy_backward_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::copy_n_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::copy_result**
- template<typename _Iter1, typename _Iter2>
  using **std::ranges::move_backward_result**
- template<typename _Iter, typename _Out>
  using **std::ranges::move_result**

Functions

- template<bool _IsMove, typename _OutIter, typename _InIter>
  constexpr void **std::ranges::__detail::__assign_one** (_OutIter &__out, _InIter &__in)
- template<bool _IsMove, input_iterator _Iter, sentinel_for< _Iter > _Sent, weakly_incrementable _Out>
  constexpr __conditional_t< _IsMove, move_result< _Iter, _Out >, copy_result< _Iter, _Out > > **std::ranges::__copy_or_move** (_Iter __first, _Sent __last, _Out __result)
- template<bool _IsMove, bidirectional_iterator _Iter, sentinel_for< _Iter > _Sent, bidirectional_iterator _Out>
  constexpr __conditional_t< _IsMove, move_backward_result< _Iter, _Out >, copy_backward_result< _Iter, _Out > > **std::ranges::__copy_or_move_backward** (_Iter __first, _Sent __last, _Out __result)

Variables

- template<typename _Tp>
  constexpr bool **std::ranges::__detail::__is_move_iterator**
- template<typename _Iterator>
  constexpr bool **std::ranges::__detail::__is_move_iterator**< **move_iterator**< **_Iterator** > >
- template<typename _Tp>
  constexpr bool **std::ranges::__detail::__is_normal_iterator**
- template<typename _Iterator, typename _Container>
  constexpr bool **std::ranges::__detail::__is_normal_iterator**< **__gnu_cxx::__normal_iterator**< **_Iterator, _Container** > >
- template<typename _Tp>
  constexpr bool **std::ranges::__detail::__is_reverse_iterator**
- template<typename _Iterator>
  constexpr bool **std::ranges::__detail::__is_reverse_iterator**< **reverse_iterator**< **_Iterator** > >
- constexpr __copy_fn **std::ranges::copy**
- constexpr __copy_backward_fn **std::ranges::copy_backward**
- constexpr __copy_n_fn **std::ranges::copy_n**
- constexpr __equal_fn **std::ranges::equal**
- constexpr __fill_fn **std::ranges::fill**
- constexpr __fill_n_fn **std::ranges::fill_n**
- constexpr __move_fn **std::ranges::move**
- constexpr __move_backward_fn **std::ranges::move_backward**

7.128.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <algorithm>.

7.129 ranges_base.h File Reference

Classes

- struct std::ranges::dangling
- struct std::ranges::view_base

Namespaces

- namespace std

Concepts

- concept std::ranges::bidirectional_range
- concept std::ranges::borrowed_range
- concept std::ranges::common_range
- concept std::ranges::contiguous_range
- concept std::ranges::forward_range
- concept std::ranges::input_range
- concept std::ranges::output_range
- concept std::ranges::random_access_range
- concept std::ranges::range
- concept std::ranges::sized_range
- concept std::ranges::view
- concept std::ranges::viewable_range

Macros

- #define **_GLIBCXX26_RANGE_ALGO_DEF_VAL_T**(_I, _P)

Typedefs

- template<typename _Tp>
  using **std::ranges::__detail::__make_unsigned_like_t**
- template<typename _Tp>
  using **std::ranges::__access::__range_iter_t**
- template<range _Range>
  using **std::ranges::borrowed_iterator_t**
- template<range _Range>
  using **std::ranges::const_iterator_t**
- template<range _Range>
  using **std::ranges::const_sentinel_t**
- template<typename _Tp>
  using **std::ranges::iterator_t**
- template<range _Range>
  using **std::ranges::range_common_reference_t**
- template<range _Range>
  using **std::ranges::range_const_reference_t**
- template<range _Range>
  using **std::ranges::range_difference_t**
- template<range _Range>
  using **std::ranges::range_reference_t**
- template<range _Range>
  using **std::ranges::range_rvalue_reference_t**
- template<sized_range _Range>
  using **std::ranges::range_size_t**
- template<range _Range>
  using **std::ranges::range_value_t**
- template<range _Range>
  using **std::ranges::sentinel_t**

Functions

- template<typename _Tp, typename _Up>
  void **std::ranges::__detail::__is_derived_from_view_interface_fn** (const _Tp &, const view_interface< _Up > &)
- template<input_range _Range>
  constexpr auto & **std::ranges::__access::__possibly_const_range** (_Range &__r) noexcept
- template<__static_sized_range _Range>
  consteval range_size_t< _Range > **std::ranges::__static_size** ()
- constexpr __max_size_type **std::ranges::__detail::__to_unsigned_like** (__max_diff_type __t) noexcept
- constexpr __max_size_type **std::ranges::__detail::__to_unsigned_like** (__max_size_type __t) noexcept
- template<integral _Tp>
  constexpr auto **std::ranges::__detail::__to_unsigned_like** (_Tp __t) noexcept
- void **std::ranges::__access::end** ()=delete
- void **std::ranges::__access::rbegin** ()=delete
- void **std::ranges::__access::rend** ()=delete
- void **std::ranges::__access::size** ()=delete

Variables

- template<typename _Tp>
  constexpr bool **std::ranges::__detail::__is_initializer_list**
- template<typename _Tp>
  constexpr bool **std::ranges::__detail::__is_initializer_list**< **initializer_list**< **_Tp** > >
- constexpr __advance_fn **std::ranges::advance**
- constexpr ranges::__access::_Begin **std::ranges::begin**
- constexpr ranges::__access::_CBegin **std::ranges::cbegin**
- constexpr ranges::__access::_CData **std::ranges::cdata**
- constexpr ranges::__access::_CEnd **std::ranges::cend**
- constexpr ranges::__access::_CRBegin **std::ranges::crbegin**
- constexpr ranges::__access::_CREnd **std::ranges::crend**
- constexpr ranges::__access::_Data **std::ranges::data**
- template<typename>
  constexpr bool **std::ranges::disable_sized_range**
- constexpr __distance_fn **std::ranges::distance**
- constexpr ranges::__access::_Empty **std::ranges::empty**
- template<typename _Tp>
  constexpr bool **std::ranges::enable_borrowed_range**
- template<typename _Tp>
  constexpr bool std::ranges::enable_view
- constexpr ranges::__access::_End **std::ranges::end**
- constexpr __next_fn **std::ranges::next**
- constexpr __prev_fn **std::ranges::prev**
- constexpr ranges::__access::_RBegin **std::ranges::rbegin**
- constexpr ranges::__access::_REnd **std::ranges::rend**
- constexpr ranges::__access::_Size **std::ranges::size**
- constexpr ranges::__access::_SSize **std::ranges::ssize**

7.129.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ranges>.

7.129.2 Variable Documentation

enable_view

```
template<typename _Tp>
bool std::ranges::enable_view  [inline], [constexpr]
```

[range.view] The ranges::enable_view boolean.

7.130 ranges_cmp.h File Reference

Classes

- struct std::ranges::equal_to
- struct std::ranges::greater
- struct std::ranges::greater_equal
- struct std::identity
- struct std::ranges::less
- struct std::ranges::less_equal
- struct std::ranges::not_equal_to

Namespaces

- namespace std

7.130.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

7.131 ranges_uninitialized.h File Reference

7.131.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.132 ranges_util.h File Reference

Classes

- class std::ranges::subrange< _It, _Sent, _Kind >
- class std::ranges::view_interface< _Derived >

Namespaces

- namespace std

Typedefs

- template<range _Range>
  using **std::ranges::borrowed_subrange_t**
- template<typename _Iter1, typename _Iter2>
  using **std::ranges::mismatch_result**

Enumerations

- enum class **subrange_kind** : bool { **unsized** , **sized** }

Functions

- template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>
  constexpr auto **std::get** (const subrange< _It, _Sent, _Kind > &__r)
- template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>
  constexpr auto **std::ranges::get** (const subrange< _It, _Sent, _Kind > &__r)
- template<size_t _Num, class _It, class _Sent, subrange_kind _Kind>
  constexpr auto **std::ranges::get** (subrange< _It, _Sent, _Kind > &&__r)
- template<input_or_output_iterator _It, sentinel_for< _It > _Sent>
  std::ranges::subrange (_It, _Sent) -> subrange< _It, _Sent >
- template<input_or_output_iterator _It, sentinel_for< _It > _Sent>
  std::ranges::subrange (_It, _Sent, __detail::__make_unsigned_like_t< iter_difference_t< _It > >) -> subrange< _It, _Sent, subrange_kind::sized >
- template<borrowed_range _Rng>
  std::ranges::subrange (_Rng &&) -> subrange< iterator_t< _Rng >, sentinel_t< _Rng >,(sized_range< _Rng >||sized_sentinel_for< sentinel_t< _Rng >, iterator_t< _Rng > >) ? subrange_kind::sized :subrange_kind::↩ unsized >
- template<borrowed_range _Rng>
  std::ranges::subrange (_Rng &&, __detail::__make_unsigned_like_t< range_difference_t< _Rng > >) -> subrange< iterator_t< _Rng >, sentinel_t< _Rng >, subrange_kind::sized >

Variables

- template<typename _It, typename _Sent, ranges::subrange_kind _Kind>
  constexpr bool **std::__is_tuple_like_v< ranges::subrange< _It, _Sent, _Kind > >**
- constexpr __adjacent_find_fn **std::ranges::adjacent_find**
- template<typename _It, typename _Sent, subrange_kind _Kind>
  constexpr bool **std::ranges::enable_borrowed_range< subrange< _It, _Sent, _Kind > >**
- constexpr __find_fn **std::ranges::find**
- constexpr __find_if_fn **std::ranges::find_if**
- constexpr __find_if_not_fn **std::ranges::find_if_not**
- constexpr __min_fn **std::ranges::min**
- constexpr __mismatch_fn **std::ranges::mismatch**
- constexpr __search_fn **std::ranges::search**

7.132.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ranges>.

7.133 refwrap.h File Reference

Classes

- class std::reference_wrapper< _Tp >

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Tp>
  std::reference_wrapper (_Tp &) -> reference_wrapper< _Tp >

Variables

- template<typename _Tp>
  constexpr bool **std::__detail::__is_ref_wrapper**
- template<typename _Tp>
  constexpr bool **std::__detail::__is_ref_wrapper**< **reference_wrapper**< **_Tp** > >

7.133.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

7.134 regex.h File Reference

Classes

- class std::basic_regex< _Ch_type, _Rx_traits >
- class std::match_results< _Bi_iter, _Alloc >
- class std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >
- class std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >
- class std::regex_traits< _Ch_type >
- class std::sub_match< _BiIter >

Namespaces

- namespace std
- namespace std::__detail

Typedefs

- typedef match_results< const char * > **std::cmatch**
- typedef regex_iterator< const char * > **std::cregex_iterator**
- typedef regex_token_iterator< const char * > std::cregex_token_iterator
- typedef sub_match< const char * > std::csub_match
- typedef basic_regex< char > std::regex
- typedef match_results< string::const_iterator > **std::smatch**
- typedef regex_iterator< string::const_iterator > **std::sregex_iterator**
- typedef regex_token_iterator< string::const_iterator > std::sregex_token_iterator
- typedef sub_match< string::const_iterator > std::ssub_match
- typedef match_results< const wchar_t * > **std::wcmatch**
- typedef regex_iterator< const wchar_t * > **std::wcregex_iterator**
- typedef regex_token_iterator< const wchar_t * > std::wcregex_token_iterator
- typedef sub_match< const wchar_t * > std::wcsub_match
- typedef basic_regex< wchar_t > std::wregex
- typedef match_results< wstring::const_iterator > **std::wsmatch**
- typedef regex_iterator< wstring::const_iterator > **std::wsregex_iterator**
- typedef regex_token_iterator< wstring::const_iterator > std::wsregex_token_iterator
- typedef sub_match< wstring::const_iterator > std::wssub_match

Enumerations

- enum class **_RegexExecutorPolicy** : int { **_S_auto** , **_S_alternate** }

Functions

- template<typename _BiIter, typename _Alloc, typename _CharT, typename _TraitsT>
  bool **std::__detail::__regex_algo_impl** (_BiIter __s, _BiIter __e, match_results< _BiIter, _Alloc > &__m, const basic_regex< _CharT, _TraitsT > &__re, regex_constants::match_flag_type __flags, _RegexExecutorPolicy _↩
  _policy, bool __match_mode)
- template<typename _ForwardIterator>
  std::basic_regex (_ForwardIterator, _ForwardIterator, regex_constants::syntax_option_type={}) -> basic_↩
  regex< typename iterator_traits< _ForwardIterator >::value_type >

Matching, Searching, and Replacing

- template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type, class _Rx_traits>
  bool std::regex_match (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_type, typename _Alloc, typename _Rx_traits>
  bool std::regex_match (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &↩
  __m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __↩
  f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_traits, typename _Str_allocator, typename _Ch_type, typename _Rx_traits>
  bool std::regex_match (const basic_string< _Ch_type, _Ch_traits, _Str_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
  _Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  _Out_iter std::regex_replace (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type>
  basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_↩
  traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  basic_string< _Ch_type > std::regex_replace (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx↩
  _traits > &__e, const basic_string< _Ch_type, _St, _Sa > &__fmt, regex_constants::match_flag_type __↩
  flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa>
  basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Rx_traits, typename _Ch_type, typename _St, typename _Sa, typename _Fst, typename _Fsa>
  basic_string< _Ch_type, _St, _Sa > std::regex_replace (const basic_string< _Ch_type, _St, _Sa > &__↩

s, const basic_regex< _Ch_type, _Rx_traits > &__e, const basic_string< _Ch_type, _Fst, _Fsa > &__fmt, regex_constants::match_flag_type __flags=regex_constants::match_default)

- template<typename _Bi_iter, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (_Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Bi_iter, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (_Bi_iter __s, _Bi_iter __e, match_results< _Bi_iter, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__re, regex_constants::match_flag_type __flags=regex_constants::match_default)
- template<typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const _Ch_type *__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_type, class _Alloc, class _Rx_traits>
  bool std::regex_search (const _Ch_type *__s, match_results< const _Ch_type *, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &&, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &, const basic_regex< _Ch_type, _Rx_traits > &, regex_constants::match_flag_type=regex_constants::match_default)=delete
- template<typename _Ch_traits, typename _Ch_alloc, typename _Alloc, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _Ch_alloc > &__s, match_results< typename basic_string< _Ch_type, _Ch_traits, _Ch_alloc >::const_iterator, _Alloc > &__m, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __f=regex_constants::match_default)
- template<typename _Ch_traits, typename _String_allocator, typename _Ch_type, typename _Rx_traits>
  bool std::regex_search (const basic_string< _Ch_type, _Ch_traits, _String_allocator > &__s, const basic_regex< _Ch_type, _Rx_traits > &__e, regex_constants::match_flag_type __flags=regex_constants::match_default)

7.134.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

7.135 regex.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Out_iter, typename _Bi_iter, typename _Rx_traits, typename _Ch_type>
  _Out_iter **std::__regex_replace** (_Out_iter __out, _Bi_iter __first, _Bi_iter __last, const basic_regex< _Ch_type, _Rx_traits > &__e, const _Ch_type *__fmt, size_t __len, regex_constants::match_flag_type __flags)

7.135.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

7.136 regex_automaton.h File Reference

Classes

- class std::__detail::_StateSeq< _TraitsT >

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_REGEX_STATE_LIMIT**

Typedefs

- template<typename _CharT>
  using **std::__detail::_Matcher**
- typedef long **std::__detail::_StateIdT**

Enumerations

- enum std::__detail::_Opcode : int {
  _S_opcode_unknown , **_S_opcode_alternative** , **_S_opcode_repeat** , **_S_opcode_backref** ,
  _S_opcode_line_begin_assertion , **_S_opcode_line_end_assertion** , **_S_opcode_word_boundary** , **_S_↩opcode_subexpr_lookahead** ,
  _S_opcode_subexpr_begin , **_S_opcode_subexpr_end** , **_S_opcode_dummy** , **_S_opcode_match** ,
  _S_opcode_accept }

Variables

- constexpr _StateIdT **std::__detail::_S_invalid_state_id**

7.136.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

7.137 regex_automaton.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

7.137.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

7.138 regex_compiler.h File Reference

Classes

- struct std::__detail::_BracketMatcher< _TraitsT, __icase, __collate >
- class std::__detail::_Compiler< _TraitsT >

Namespaces

- namespace std
- namespace std::__detail

7.138.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

7.139 regex_compiler.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **__INSERT_REGEX_MATCHER**(__func, ...)

7.139.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

7.140 regex_constants.h File Reference

Namespaces

- namespace std
- namespace std::regex_constants

5.1 Regular Expression Syntax Options

- constexpr syntax_option_type std::regex_constants::__multiline
- constexpr syntax_option_type std::regex_constants::__polynomial
- constexpr syntax_option_type std::regex_constants::awk
- constexpr syntax_option_type std::regex_constants::basic
- constexpr syntax_option_type std::regex_constants::collate
- constexpr syntax_option_type std::regex_constants::ECMAScript
- constexpr syntax_option_type std::regex_constants::egrep
- constexpr syntax_option_type std::regex_constants::extended
- constexpr syntax_option_type std::regex_constants::grep
- constexpr syntax_option_type std::regex_constants::icase
- constexpr syntax_option_type std::regex_constants::multiline
- constexpr syntax_option_type std::regex_constants::nosubs
- constexpr syntax_option_type std::regex_constants::operator& (syntax_option_type __a, syntax_option_type _↩ _b) noexcept
- constexpr syntax_option_type & std::regex_constants::operator&= (syntax_option_type &__a, syntax_option_type __b) noexcept
- constexpr syntax_option_type std::regex_constants::operator^ (syntax_option_type __a, syntax_option_type _↩ _b) noexcept
- constexpr syntax_option_type & std::regex_constants::operator^= (syntax_option_type &__a, syntax_option_type __b) noexcept
- constexpr syntax_option_type std::regex_constants::operator| (syntax_option_type __a, syntax_option_type __b) noexcept
- constexpr syntax_option_type & std::regex_constants::operator|= (syntax_option_type &__a, syntax_option_type __b) noexcept

- constexpr syntax_option_type std::regex_constants::operator~ (syntax_option_type __a) noexcept
- constexpr syntax_option_type std::regex_constants::optimize
- enum std::regex_constants::syntax_option_type : unsigned int {
  _S_icase , **_S_nosubs** , **_S_optimize** , **_S_collate** ,
  _S_ECMAScript , **_S_basic** , **_S_extended** , **_S_awk** ,
  _S_grep , **_S_egrep** , **_S_polynomial** , **_S_multiline** }

5.2 Matching Rules

Matching a regular expression against a sequence of characters [first, last) proceeds according to the rules of the grammar specified for the regular expression object, modified according to the effects listed below for any bitmask elements set.

- constexpr match_flag_type std::regex_constants::format_default
- constexpr match_flag_type std::regex_constants::format_first_only
- constexpr match_flag_type std::regex_constants::format_no_copy
- constexpr match_flag_type std::regex_constants::format_sed
- constexpr match_flag_type std::regex_constants::match_any
- constexpr match_flag_type std::regex_constants::match_continuous
- constexpr match_flag_type std::regex_constants::match_default
- enum std::regex_constants::match_flag_type : unsigned int {
  _S_default , **_S_not_bol** , **_S_not_eol** , **_S_not_bow** ,
  _S_not_eow , **_S_any** , **_S_not_null** , **_S_continuous** ,
  _S_prev_avail , **_S_sed** , **_S_no_copy** , **_S_first_only** ,
  _S_match_flag_last }
- constexpr match_flag_type std::regex_constants::match_not_bol
- constexpr match_flag_type std::regex_constants::match_not_bow
- constexpr match_flag_type std::regex_constants::match_not_eol
- constexpr match_flag_type std::regex_constants::match_not_eow
- constexpr match_flag_type std::regex_constants::match_not_null
- constexpr match_flag_type std::regex_constants::match_prev_avail
- constexpr match_flag_type std::regex_constants::operator& (match_flag_type __a, match_flag_type __b) noexcept
- constexpr match_flag_type & std::regex_constants::operator&= (match_flag_type &__a, match_flag_type __b) noexcept
- constexpr match_flag_type std::regex_constants::operator^ (match_flag_type __a, match_flag_type __b) noexcept
- constexpr match_flag_type & std::regex_constants::operator^= (match_flag_type &__a, match_flag_type __b) noexcept
- constexpr match_flag_type std::regex_constants::operator| (match_flag_type __a, match_flag_type __b) noexcept
- constexpr match_flag_type & std::regex_constants::operator|= (match_flag_type &__a, match_flag_type __↩ b) noexcept
- constexpr match_flag_type std::regex_constants::operator~ (match_flag_type __a) noexcept

7.140.1 Detailed Description

Constant definitions for the std regex library.
This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

7.141 regex_error.h File Reference

Classes

- class std::regex_error

Namespaces

- namespace std
- namespace std::regex_constants

5.3 Error Types

- constexpr error_type std::regex_constants::error_backref (_S_error_backref)
- constexpr error_type std::regex_constants::error_badbrace (_S_error_badbrace)
- constexpr error_type std::regex_constants::error_badrepeat (_S_error_badrepeat)
- constexpr error_type std::regex_constants::error_brace (_S_error_brace)
- constexpr error_type std::regex_constants::error_brack (_S_error_brack)
- constexpr error_type std::regex_constants::error_collate (_S_error_collate)
- constexpr error_type std::regex_constants::error_complexity (_S_error_complexity)
- constexpr error_type std::regex_constants::error_ctype (_S_error_ctype)
- constexpr error_type std::regex_constants::error_escape (_S_error_escape)
- constexpr error_type std::regex_constants::error_paren (_S_error_paren)
- constexpr error_type std::regex_constants::error_range (_S_error_range)
- constexpr error_type std::regex_constants::error_space (_S_error_space)
- constexpr error_type std::regex_constants::error_stack (_S_error_stack)
- enum std::regex_constants::error_type {
  _S_error_collate , **_S_error_ctype** , **_S_error_escape** , **_S_error_backref** ,
  _S_error_brack , **_S_error_paren** , **_S_error_brace** , **_S_error_badbrace** ,
  _S_error_range , **_S_error_space** , **_S_error_badrepeat** , **_S_error_complexity** ,
  _S_error_stack , **_S_null** , **_S_grammar** }

7.141.1 Detailed Description

Error and exception objects for the std regex library.
This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

7.142 regex_executor.h File Reference

Classes

- class std::__detail::_Executor< _BiIter, _Alloc, _TraitsT >

Namespaces

- namespace std
- namespace std::__detail

7.142.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

7.143 regex_executor.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

Enumerations

- enum **_ExecutorFrameOpcode** : unsigned char {
  _S_fopcode_next , **_S_fopcode_fallback_next** , **_S_fopcode_rep_once_more** , **_S_fopcode_fallback_rep_once_more** ,
  _S_fopcode_posix_alternative , **_S_fopcode_merge_sol** , **_S_fopcode_restore_cur_results** , **_S_fopcode_restore_rep_count** ,
  _S_fopcode_decrement_rep_count }

7.143.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

7.144 regex_scanner.h File Reference

Classes

- class std::__detail::_Scanner< _CharT >

Namespaces

- namespace std
- namespace std::__detail

7.144.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

7.145 regex_scanner.tcc File Reference

Namespaces

- namespace std
- namespace std::__detail

7.145.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <regex>.

7.146 requires_hosted.h File Reference

7.146.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <version>.

7.147 sat_arith.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Tp>
  constexpr _Tp std::saturating_add (_Tp __x, _Tp __y) noexcept
- template<typename _Res, typename _Tp>
  constexpr _Res std::saturating_cast (_Tp __x) noexcept
- template<typename _Tp>
  constexpr _Tp std::saturating_div (_Tp __x, _Tp __y) noexcept
- template<typename _Tp>
  constexpr _Tp std::saturating_mul (_Tp __x, _Tp __y) noexcept
- template<typename _Tp>
  constexpr _Tp std::saturating_sub (_Tp __x, _Tp __y) noexcept

7.147.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <numeric>.

7.148 semaphore_base.h File Reference

7.148.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <semaphore>.

7.149 bits/shared_ptr.h File Reference

Classes

- class std::enable_shared_from_this< _Tp >
- struct std::hash< shared_ptr< _Tp > >
- struct std::owner_less< shared_ptr< _Tp > >
- struct std::owner_less< void >
- struct std::owner_less< weak_ptr< _Tp > >
- class std::shared_ptr< _Tp >
- class std::weak_ptr< _Tp >

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Del, typename _Tp, _Lock_policy _Lp>
  _Del * **std::get_deleter** (const __shared_ptr< _Tp, _Lp > &__p) noexcept
- template<typename _Tp, typename _Del>
  std::shared_ptr (unique_ptr< _Tp, _Del >) -> shared_ptr< _Tp >
- template<typename _Tp>
  std::shared_ptr (weak_ptr< _Tp >) -> shared_ptr< _Tp >

- template<typename _Tp>
  std::weak_ptr (shared_ptr< _Tp >) -> weak_ptr< _Tp >

7.149.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.150 experimental/bits/shared_ptr.h File Reference

Classes

- struct std::hash< experimental::shared_ptr< _Tp > >
- struct std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >
- struct std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >

Namespaces

- namespace std
- namespace std::experimental

Functions

- template<typename _Tp>
  bool **std::experimental::atomic_compare_exchange_strong** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp>
  bool **std::experimental::atomic_compare_exchange_strong_explicit** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  bool **std::experimental::atomic_compare_exchange_weak** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp>
  bool **std::experimental::atomic_compare_exchange_weak_explicit** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  void **std::experimental::atomic_exchange** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp>
  shared_ptr< _Tp > **std::experimental::atomic_exchange_explicit** (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)
- template<typename _Tp>
  bool **std::experimental::atomic_is_lock_free** (const shared_ptr< _Tp > *__p)
- template<typename _Tp>
  shared_ptr< _Tp > **std::experimental::atomic_load** (const shared_ptr< _Tp > *__p)
- template<typename _Tp>
  shared_ptr< _Tp > **std::experimental::atomic_load_explicit** (const shared_ptr< _Tp > *__p, memory_order __mo)
- template<typename _Tp>
  void **std::experimental::atomic_store** (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp>
  shared_ptr< _Tp > **std::experimental::atomic_store_explicit** (const shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order __mo)
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **std::experimental::const_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept

- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **std::experimental::dynamic_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Del, typename _Tp>
  _Del * std::experimental::get_deleter (const shared_ptr< _Tp > &__p) noexcept
- template<typename _Tp>
  bool **std::experimental::operator!=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::experimental::operator!=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **std::experimental::operator!=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp>
  bool **std::experimental::operator<** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::experimental::operator<** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **std::experimental::operator<** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Ch, typename _Tr, typename _Tp>
  std::basic_ostream< _Ch, _Tr > & **std::experimental::operator<<** (std::basic_ostream< _Ch, _Tr > &__os, const shared_ptr< _Tp > &__p)
- template<typename _Tp>
  bool **std::experimental::operator<=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::experimental::operator<=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **std::experimental::operator<=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp>
  bool **std::experimental::operator==** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::experimental::operator==** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **std::experimental::operator==** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp>
  bool **std::experimental::operator>** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::experimental::operator>** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **std::experimental::operator>** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp>
  bool **std::experimental::operator>=** (const shared_ptr< _Tp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2>
  bool **std::experimental::operator>=** (const shared_ptr< _Tp1 > &__a, const shared_ptr< _Tp2 > &__b) noexcept
- template<typename _Tp>
  bool **std::experimental::operator>=** (nullptr_t, const shared_ptr< _Tp > &__a) noexcept
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **std::experimental::reinterpret_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept
- template<typename _Tp, typename _Tp1>
  shared_ptr< _Tp > **std::experimental::static_pointer_cast** (const shared_ptr< _Tp1 > &__r) noexcept

- template<typename _Tp>
  void **std::experimental::swap** (shared_ptr< _Tp > &__a, shared_ptr< _Tp > &__b) noexcept
- template<typename _Tp>
  void **std::experimental::swap** (weak_ptr< _Tp > &__a, weak_ptr< _Tp > &__b) noexcept

Variables

- template<typename _Yp, typename _Tp>
  constexpr bool **std::experimental::__sp_compatible_v**
- template<typename _Tp, typename _Yp>
  constexpr bool **std::experimental::__sp_is_constructible_v**

7.150.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <experimental/memory>.

7.151 shared_ptr_atomic.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_TSAN_MUTEX_DESTROY**(X)
- #define **_GLIBCXX_TSAN_MUTEX_LOCKED**(X)
- #define **_GLIBCXX_TSAN_MUTEX_POST_SIGNAL**(X)
- #define **_GLIBCXX_TSAN_MUTEX_POST_UNLOCK**(X)
- #define **_GLIBCXX_TSAN_MUTEX_PRE_SIGNAL**(X)
- #define **_GLIBCXX_TSAN_MUTEX_PRE_UNLOCK**(X)
- #define **_GLIBCXX_TSAN_MUTEX_TRY_LOCK**(X)
- #define **_GLIBCXX_TSAN_MUTEX_TRY_LOCK_FAILED**(X)

Functions

- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_is_lock_free (const __shared_ptr< _Tp, _Lp > *)
- template<typename _Tp>
  bool std::atomic_is_lock_free (const shared_ptr< _Tp > *__p)

- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_load (const __shared_ptr< _Tp, _Lp > *__p)
- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_load (const shared_ptr< _Tp > *__p)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_load_explicit (const __shared_ptr< _Tp, _Lp > *__p, memory_order)
- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_load_explicit (const shared_ptr< _Tp > *__p, memory_order)

- template<typename _Tp, _Lock_policy _Lp>
  void std::atomic_store (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)
- template<typename _Tp>
  void std::atomic_store (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp, _Lock_policy _Lp>
  void std::atomic_store_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)
- template<typename _Tp>
  void std::atomic_store_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)

- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_exchange (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r)
- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_exchange (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r)
- template<typename _Tp, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::atomic_exchange_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > __r, memory_order)
- template<typename _Tp>
  shared_ptr< _Tp > std::atomic_exchange_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > __r, memory_order)

- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_strong (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)
- template<typename _Tp>
  bool std::atomic_compare_exchange_strong (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_strong_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order, memory_order)
- template<typename _Tp>
  bool std::atomic_compare_exchange_strong_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order, memory_order)
- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_weak (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w)
- template<typename _Tp>
  bool std::atomic_compare_exchange_weak (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w)
- template<typename _Tp, _Lock_policy _Lp>
  bool std::atomic_compare_exchange_weak_explicit (__shared_ptr< _Tp, _Lp > *__p, __shared_ptr< _Tp, _Lp > *__v, __shared_ptr< _Tp, _Lp > __w, memory_order __success, memory_order __failure)
- template<typename _Tp>
  bool std::atomic_compare_exchange_weak_explicit (shared_ptr< _Tp > *__p, shared_ptr< _Tp > *__v, shared_ptr< _Tp > __w, memory_order __success, memory_order __failure)

7.151.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.152 shared_ptr_base.h File Reference

Classes

- struct std::_Sp_ebo_helper< _Tp, false >
- struct std::_Sp_ebo_helper< _Tp, true >
- class std::bad_weak_ptr
- struct std::hash< __shared_ptr< _Tp, _Lp > >

Namespaces

- namespace std

Enumerations

- enum **_Lock_policy**

Functions

- template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename _Alloc, typename... _Args>
  __shared_ptr< _Tp, _Lp > **std::__allocate_shared** (const _Alloc &__a, _Args &&... __args)
- template<typename _Tp, _Lock_policy _Lp = __default_lock_policy, typename... _Args>
  __shared_ptr< _Tp, _Lp > **std::__make_shared** (_Args &&... __args)
- template<typename _Tp>
  _Tp * **std::__shared_ptr_deref** (_Tp *__p)
- void **std::__throw_bad_weak_ptr** ()
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::const_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::dynamic_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, typename _Up, _Lock_policy _Lp>
  strong_ordering **std::operator<=>** (const __shared_ptr< _Tp, _Lp > &__a, const __shared_ptr< _Up, _Lp > &__b) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  strong_ordering **std::operator<=>** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  bool **std::operator==** (const __shared_ptr< _Tp, _Lp > &__a, nullptr_t) noexcept
- template<typename _Tp1, typename _Tp2, _Lock_policy _Lp>
  bool **std::operator==** (const __shared_ptr< _Tp1, _Lp > &__a, const __shared_ptr< _Tp2, _Lp > &__b) noexcept
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > **std::reinterpret_pointer_cast** (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, typename _Tp1, _Lock_policy _Lp>
  __shared_ptr< _Tp, _Lp > std::static_pointer_cast (const __shared_ptr< _Tp1, _Lp > &__r) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  void **std::swap** (__shared_ptr< _Tp, _Lp > &__a, __shared_ptr< _Tp, _Lp > &__b) noexcept
- template<typename _Tp, _Lock_policy _Lp>
  void **std::swap** (__weak_ptr< _Tp, _Lp > &__a, __weak_ptr< _Tp, _Lp > &__b) noexcept

Variables

- const _Lock_policy **std::__default_lock_policy**

7.152.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.153 slice_array.h File Reference

Classes

- class std::slice
- class std::slice_array< _Tp >

Namespaces

- namespace std

7.153.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

7.154 specfun.h File Reference

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_ai (_Tp __x)
- float __gnu_cxx::airy_aif (float __x)
- long double __gnu_cxx::airy_ail (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type __gnu_cxx::airy_bi (_Tp __x)
- float __gnu_cxx::airy_bif (float __x)
- long double __gnu_cxx::airy_bil (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float std::assoc_laguerref (unsigned int __n, unsigned int __m, float __x)
- long double std::assoc_laguerrel (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float std::assoc_legendref (unsigned int __l, unsigned int __m, float __x)
- long double std::assoc_legendrel (unsigned int __l, unsigned int __m, long double __x)
- template<typename _Tpa, typename _Tpb>
  __gnu_cxx::__promote_2< _Tpa, _Tpb >::__type std::beta (_Tpa __a, _Tpb __b)
- float std::betaf (float __a, float __b)
- long double std::betal (long double __a, long double __b)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::comp_ellint_1 (_Tp __k)
- float std::comp_ellint_1f (float __k)
- long double std::comp_ellint_1l (long double __k)

- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::comp_ellint_2 (_Tp __k)
- float std::comp_ellint_2f (float __k)
- long double std::comp_ellint_2l (long double __k)
- template<typename _Tp, typename _Tpn>
  __gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::comp_ellint_3 (_Tp __k, _Tpn __nu)
- float std::comp_ellint_3f (float __k, float __nu)
- long double std::comp_ellint_3l (long double __k, long double __nu)
- template<typename _Tpa, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type __gnu_cxx::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)
- float __gnu_cxx::conf_hypergf (float __a, float __c, float __x)
- long double __gnu_cxx::conf_hypergl (long double __a, long double __c, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float std::cyl_bessel_if (float __nu, float __x)
- long double std::cyl_bessel_il (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float std::cyl_bessel_jf (float __nu, float __x)
- long double std::cyl_bessel_jl (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float std::cyl_bessel_kf (float __nu, float __x)
- long double std::cyl_bessel_kl (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::cyl_neumann (_Tpnu __nu, _Tp __x)
- float std::cyl_neumannf (float __nu, float __x)
- long double std::cyl_neumannl (long double __nu, long double __x)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_1 (_Tp __k, _Tpp __phi)
- float std::ellint_1f (float __k, float __phi)
- long double std::ellint_1l (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::ellint_2 (_Tp __k, _Tpp __phi)
- float std::ellint_2f (float __k, float __phi)
- long double std::ellint_2l (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp>
  __gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float std::ellint_3f (float __k, float __nu, float __phi)
- long double std::ellint_3l (long double __k, long double __nu, long double __phi)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::expint (_Tp __x)
- float std::expintf (float __x)
- long double std::expintl (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::hermite (unsigned int __n, _Tp __x)
- float std::hermitef (unsigned int __n, float __x)
- long double std::hermitel (unsigned int __n, long double __x)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type __gnu_cxx::hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float __gnu_cxx::hypergf (float __a, float __b, float __c, float __x)

- long double __gnu_cxx::hypergl (long double __a, long double __b, long double __c, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::laguerre (unsigned int __n, _Tp __x)
- float std::laguerref (unsigned int __n, float __x)
- long double std::laguerrel (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::legendre (unsigned int __l, _Tp __x)
- float std::legendref (unsigned int __l, float __x)
- long double std::legendrel (unsigned int __l, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::riemann_zeta (_Tp __s)
- float std::riemann_zetaf (float __s)
- long double std::riemann_zetal (long double __s)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::sph_bessel (unsigned int __n, _Tp __x)
- float std::sph_besself (unsigned int __n, float __x)
- long double std::sph_bessell (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float std::sph_legendref (unsigned int __l, unsigned int __m, float __theta)
- long double std::sph_legendrel (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::sph_neumann (unsigned int __n, _Tp __x)
- float std::sph_neumannf (unsigned int __n, float __x)
- long double std::sph_neumannl (unsigned int __n, long double __x)

7.154.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <cmath>.

7.155 sstream.tcc File Reference

Namespaces

- namespace std

7.155.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <sstream>.

7.156 std_abs.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Tp>
  _Tp std::abs (const complex< _Tp > &)
- constexpr double **std::abs** (double __x)
- constexpr float **std::abs** (float __x)

- long **std::abs** (long __i)
- constexpr long double **std::abs** (long double __x)
- long long **std::abs** (long long __x)

7.156.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <cmath> or < cstdlib>.

7.157 std_function.h File Reference

Classes

- struct std::__is_location_invariant< _Tp >
- class std::_Function_base
- class std::bad_function_call
- class std::function< _Res(_ArgTypes...)>

Namespaces

- namespace std
- namespace std::__detail

Typedefs

- template<typename _Fn, typename _Op>
  using **std::__function_guide_t**

Enumerations

- enum **_Manager_operation** { **__get_type_info** , **__get_functor_ptr** , **__clone_functor** , **__destroy_functor** }

Functions

- template<typename _Fn, typename _Signature = __function_guide_t<_Fn, decltype(&_Fn::operator())>>
  std::function (_Fn) -> function< _Signature >
- template<typename _Res, typename... _ArgTypes>
  std::function (_Res(*)(_ArgTypes...)) -> function< _Res(_ArgTypes...)>
- template<typename _Res, typename... _Args>
  bool std::operator== (const function< _Res(_Args...)> &__f, nullptr_t) noexcept
- template<typename _Res, typename... _Args>
  void std::swap (function< _Res(_Args...)> &__x, function< _Res(_Args...)> &__y) noexcept

7.157.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

7.158 std_mutex.h File Reference

Classes

- struct std::adopt_lock_t
- struct std::defer_lock_t
- class std::lock_guard< _Mutex >

- class std::mutex
- struct std::try_to_lock_t

Namespaces

- namespace std

Variables

- constexpr adopt_lock_t std::adopt_lock
- constexpr defer_lock_t std::defer_lock
- constexpr try_to_lock_t std::try_to_lock

7.158.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <mutex>.

7.159 std_thread.h File Reference

Classes

- struct std::hash< thread::id >
- class std::thread::id
- class std::thread

Namespaces

- namespace std
- namespace std::this_thread

Functions

- thread::id std::this_thread::get_id () noexcept
- void std::this_thread::yield () noexcept

7.159.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <thread>.

7.160 stdexcept_except.h File Reference

Classes

- class std::domain_error
- class std::invalid_argument
- class std::length_error
- class std::logic_error
- class std::out_of_range
- class std::overflow_error
- class std::runtime_error
- class std::underflow_error

Namespaces

- namespace std

Typedefs

- typedef basic_string< char > **std::__sso_string**

7.160.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <stdexcept>.

7.161 stdexcept_throw.h File Reference

7.161.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <stdexcept>.

7.162 stdexcept_throwdef.h File Reference

7.162.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <stdexcept>.

7.163 stdexcept_throwfwd.h File Reference

Namespaces

- namespace std

Functions

- void **std::__throw_domain_error** (const char *)
- void **std::__throw_invalid_argument** (const char *)
- void **std::__throw_length_error** (const char *)
- void **std::__throw_logic_error** (const char *)
- void **std::__throw_out_of_range** (const char *)
- void **std::__throw_out_of_range_fmt** (const char *,...)
- void **std::__throw_overflow_error** (const char *)
- void **std::__throw_runtime_error** (const char *)
- void **std::__throw_underflow_error** (const char *)

7.163.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <stdexcept>.

7.164 stl_algo.h File Reference

Namespaces

- namespace std

Functions

- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator **std::__adjacent_find** (_ForwardIterator __first, _ForwardIterator __last, _Binary↩
  Predicate __binary_pred)
- template<typename _IntType, typename _UniformRandomBitGenerator>
  pair< _IntType, _IntType > std::__gen_two_uniform_ints (_IntType __b0, _IntType __b1, _UniformRandomBit↩
  Generator &&__g)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator **std::__max_element** (_ForwardIterator __first, _ForwardIterator __last, _Compare
  __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **std::__merge** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2,
  _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator **std::__min_element** (_ForwardIterator __first, _ForwardIterator __last, _Compare
  __comp)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
  constexpr _OutputIterator **std::__remove_copy_if** (_InputIterator __first, _InputIterator __last, _OutputIterator
  __result, _Predicate __pred)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **std::__set_difference** (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2
  __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **std::__set_intersection** (_InputIterator1 __first1, _InputIterator1 __last1, _Input↩
  Iterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator **std::__set_symmetric_difference** (_InputIterator1 __first1, _InputIterator1 __last1, ↩
  _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **std::__stable_sort** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare
  __comp)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator **std::__unique** (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate
  __binary_pred)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator std::adjacent_find (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate
  __binary_pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool std::all_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool std::any_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _ForwardIterator, typename _Tp>
  constexpr bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr bool std::binary_search (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare
  __comp)
- template<typename _Tp>
  constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::clamp (const _Tp &__val, const _Tp &__lo, const _Tp &__hi, _Compare __comp)

- template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
  constexpr _OutputIterator std::copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _↩
  Predicate __pred)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  constexpr _OutputIterator std::copy_n (_InputIterator __first, _Size __n, _OutputIterator __result)
- template<typename _InputIterator, typename _Tp>
  constexpr iterator_traits< _InputIterator >::difference_type std::count (_InputIterator __first, _InputIterator __last, const _Tp &__value)
- template<typename _InputIterator, typename _Predicate>
  constexpr iterator_traits< _InputIterator >::difference_type std::count_if (_InputIterator __first, _InputIterator _↩
  _last, _Predicate __pred)
- template<typename _ForwardIterator, typename _Tp>
  constexpr pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _Forward↩
  Iterator __last, const _Tp &__val)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr pair< _ForwardIterator, _ForwardIterator > std::equal_range (_ForwardIterator __first, _Forward↩
  Iterator __last, const _Tp &__val, _Compare __comp)
- template<typename _InputIterator, typename _Tp>
  constexpr _InputIterator std::find (_InputIterator __first, _InputIterator __last, const _Tp &__val)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _Forward↩
  Iterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 std::find_end (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _Forward↩
  Iterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __comp)
- template<typename _InputIterator, typename _ForwardIterator>
  constexpr _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2)
- template<typename _InputIterator, typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _InputIterator std::find_first_of (_InputIterator __first1, _InputIterator __last1, _ForwardIterator __first2, _ForwardIterator __last2, _BinaryPredicate __comp)
- template<typename _InputIterator, typename _Predicate>
  constexpr _InputIterator std::find_if (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr _InputIterator std::find_if_not (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _Function>
  constexpr _Function std::for_each (_InputIterator __first, _InputIterator __last, _Function __f)
- template<typename _InputIterator, typename _Size, typename _Function>
  constexpr _InputIterator std::for_each_n (_InputIterator __first, _Size __n, _Function __f)
- template<typename _ForwardIterator, typename _Generator>
  constexpr void std::generate (_ForwardIterator __first, _ForwardIterator __last, _Generator __gen)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  constexpr _OutputIterator std::generate_n (_OutputIterator __first, _Size __n, _Generator __gen)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↩
  Iterator2 __last2)
- template<typename _InputIterator1, typename _InputIterator2, typename _Compare>
  constexpr bool std::includes (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _Input↩
  Iterator2 __last2, _Compare __comp)
- template<typename _BidirectionalIterator>
  constexpr void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _Bidirectional↩
  Iterator __last)

- template<typename _BidirectionalIterator, typename _Compare>
  constexpr void std::inplace_merge (_BidirectionalIterator __first, _BidirectionalIterator __middle, _BidirectionalIterator __last, _Compare __comp)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool std::is_partitioned (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 _first2, _BinaryPredicate __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 _first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 _first2, _ForwardIterator2 __last2, _BinaryPredicate __pred)
- template<typename _ForwardIterator>
  constexpr bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr bool std::is_sorted (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator std::is_sorted_until (_ForwardIterator __first, _ForwardIterator __last, _Compare _comp)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _Tp>
  constexpr _Tp **std::max** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr _Tp **std::max** (initializer_list< _Tp >, _Compare)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator std::max_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::merge (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _Tp>
  constexpr _Tp **std::min** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr _Tp **std::min** (initializer_list< _Tp >, _Compare)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr _ForwardIterator std::min_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _Tp>
  constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b)

- template<typename _Tp, typename _Compare>
  constexpr pair< const _Tp &, const _Tp & > std::minmax (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp>
  constexpr pair< _Tp, _Tp > **std::minmax** (initializer_list< _Tp >)
- template<typename _Tp, typename _Compare>
  constexpr pair< _Tp, _Tp > **std::minmax** (initializer_list< _Tp >, _Compare)
- template<typename _ForwardIterator>
  constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Compare>
  constexpr pair< _ForwardIterator, _ForwardIterator > std::minmax_element (_ForwardIterator __first, _ForwardIterator __last, _Compare __comp)
- template<typename _BidirectionalIterator>
  constexpr bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool std::next_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool std::none_of (_InputIterator __first, _InputIterator __last, _Predicate __pred)
- template<typename _RandomAccessIterator>
  constexpr void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::nth_element (_RandomAccessIterator __first, _RandomAccessIterator __nth, _RandomAccessIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator>
  constexpr void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::partial_sort (_RandomAccessIterator __first, _RandomAccessIterator __middle, _RandomAccessIterator __last, _Compare __comp)
- template<typename _InputIterator, typename _RandomAccessIterator>
  constexpr _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last)
- template<typename _InputIterator, typename _RandomAccessIterator, typename _Compare>
  constexpr _RandomAccessIterator std::partial_sort_copy (_InputIterator __first, _InputIterator __last, _RandomAccessIterator __result_first, _RandomAccessIterator __result_last, _Compare __comp)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator std::partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _InputIterator, typename _OutputIterator1, typename _OutputIterator2, typename _Predicate>
  constexpr pair< _OutputIterator1, _OutputIterator2 > std::partition_copy (_InputIterator __first, _InputIterator __last, _OutputIterator1 __out_true, _OutputIterator2 __out_false, _Predicate __pred)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator std::partition_point (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _BidirectionalIterator>
  constexpr bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _Compare>
  constexpr bool std::prev_permutation (_BidirectionalIterator __first, _BidirectionalIterator __last, _Compare __comp)
- template<typename _RandomAccessIterator>
  void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last)

- template<typename _RandomAccessIterator, typename _RandomNumberGenerator>
  void std::random_shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _RandomNumber↩
  Generator &&__rand)
- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator std::remove (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator std::remove_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__value)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate>
  constexpr _OutputIterator std::remove_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __↩
  result, _Predicate __pred)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator std::remove_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void std::replace (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator std::replace_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, const _Tp &__old_value, const _Tp &__new_value)
- template<typename _InputIterator, typename _OutputIterator, typename _Predicate, typename _Tp>
  constexpr _OutputIterator std::replace_copy_if (_InputIterator __first, _InputIterator __last, _OutputIterator __↩
  result, _Predicate __pred, const _Tp &__new_value)
- template<typename _ForwardIterator, typename _Predicate, typename _Tp>
  constexpr void std::replace_if (_ForwardIterator __first, _ForwardIterator __last, _Predicate __pred, const _Tp &__new_value)
- template<typename _BidirectionalIterator>
  constexpr void std::reverse (_BidirectionalIterator __first, _BidirectionalIterator __last)
- template<typename _BidirectionalIterator, typename _OutputIterator>
  constexpr _OutputIterator std::reverse_copy (_BidirectionalIterator __first, _BidirectionalIterator __last, _Output↩
  Iterator __result)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::rotate (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator _↩
  _last)
- template<typename _ForwardIterator, typename _OutputIterator>
  constexpr _OutputIterator std::rotate_copy (_ForwardIterator __first, _ForwardIterator __middle, _ForwardIterator __last, _OutputIterator __result)
- template<typename _PopulationIterator, typename _SampleIterator, typename _Distance, typename _UniformRandomBitGenerator>
  _SampleIterator std::sample (_PopulationIterator __first, _PopulationIterator __last, _SampleIterator __out, _↩
  Distance __n, _UniformRandomBitGenerator &&__g)
- template<typename _ForwardIterator, typename _Searcher>
  constexpr _ForwardIterator std::search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &__↩
  searcher)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2)
- template<typename _ForwardIterator, typename _Integer, typename _Tp>
  constexpr _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val)
- template<typename _ForwardIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
  constexpr _ForwardIterator std::search_n (_ForwardIterator __first, _ForwardIterator __last, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 _↩
  _first2, _InputIterator2 __last2, _OutputIterator __result)

- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::set_difference (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 _↩
  _first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::set_intersection (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _↩
  InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::set_symmetric_difference (_InputIterator1 __first1, _InputIterator1 __last1, _↩
  InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator>
  constexpr _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _Compare>
  constexpr _OutputIterator std::set_union (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _OutputIterator __result, _Compare __comp)
- template<typename _RandomAccessIterator, typename _UniformRandomNumberGenerator>
  void std::shuffle (_RandomAccessIterator __first, _RandomAccessIterator __last, _UniformRandomNumber↩
  Generator &&__g)
- template<typename _RandomAccessIterator>
  constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void std::sort (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare __comp)
- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator std::stable_partition (_ForwardIterator __first, _ForwardIterator __last, _Predicate ↩
  __pred)
- template<typename _RandomAccessIterator>
  constexpr void std::stable_sort (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _InputIterator, typename _OutputIterator, typename _UnaryOperation>
  constexpr _OutputIterator std::transform (_InputIterator __first, _InputIterator __last, _OutputIterator __result, ↩
  _UnaryOperation __unary_op)
- template<typename _InputIterator1, typename _InputIterator2, typename _OutputIterator, typename _BinaryOperation>
  constexpr _OutputIterator std::transform (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _ForwardIterator>
  constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _BinaryPredicate>
  constexpr _ForwardIterator std::unique (_ForwardIterator __first, _ForwardIterator __last, _BinaryPredicate __↩
  binary_pred)
- template<typename _InputIterator, typename _OutputIterator>
  constexpr _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result)
- template<typename _InputIterator, typename _OutputIterator, typename _BinaryPredicate>
  constexpr _OutputIterator std::unique_copy (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _BinaryPredicate __binary_pred)
- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator std::upper_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__↩
  val)

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

7.164.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <algorithm>.

7.165 stl_algobase.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_ADVANCE**(P, N)
- #define **_GLIBCXX_MOVE3**(_Tp, _Up, _Vp)
- #define **_GLIBCXX_MOVE_BACKWARD3**(_Tp, _Up, _Vp)
- #define **_GLIBCXX_TO_ADDR**(P)

Functions

- template<bool _IsMove, typename _OutIter, typename _InIter>
  constexpr void **std::__assign_one** (_OutIter &__out, _InIter &__in)
- template<bool _IsMove, typename _II, typename _OI>
  constexpr _OI **std::__copy_move_a** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>
  constexpr __gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **std::__copy_move_a** (_II, _II, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &)
- template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>
  constexpr ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > **std::__copy_move_a** (const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > &)
- template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>
  constexpr _OI **std::__copy_move_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, _OI)
- template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **std::__copy_move_a1** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **std::__copy_move_a1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Tp>
  __gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value,::_Deque_iterator< _Tp, _Tp &, _Tp * > >::__type **std::__copy_move_a1** (_II __first, _II __last, ::_Deque_iterator< _Tp, _Tp &, _Tp * > __result)
- template<bool _IsMove, typename _II, typename _OI>
  constexpr _OI **std::__copy_move_a1** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type **std::__copy_move_a2** (_CharT *, _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)

- template<bool _IsMove, typename _InIter, typename _Sent, typename _OutIter>
  constexpr _OutIter **std::__copy_move_a2** (_InIter __first, _Sent __last, _OutIter __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT, char_traits< _CharT > > >::__type **std::__copy_move_a2** (const _CharT *, const _CharT *, ostreambuf_iterator< _CharT, char_traits< _CharT > >)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value,::_Deque_iterator< _CharT, _CharT &, _CharT * > >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT, char_traits< _CharT > > __first, istreambuf_iterator< _CharT, char_traits< _CharT > > __last, ::_Deque_iterator< _CharT, _CharT &, _CharT * > __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT, char_traits< _CharT > >, istreambuf_iterator< _CharT, char_traits< _CharT > >, _CharT *)
- template<bool _IsMove, typename _II, typename _OI>
  constexpr _OI **std::__copy_move_backward_a** (_II __first, _II __last, _OI __result)
- template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>
  constexpr __gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **std::__copy_move_backward_a** (_II, _II, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &)
- template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>
  constexpr ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > **std::__copy_move_backward_a** (const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > &)
- template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _OI>
  constexpr _OI **std::__copy_move_backward_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, _OI)
- template<bool _IsMove, typename _ITp, typename _IRef, typename _IPtr, typename _OTp>
  ::_Deque_iterator< _OTp, _OTp &, _OTp * > **std::__copy_move_backward_a1** (::_Deque_iterator< _ITp, _IRef, _IPtr > __first, ::_Deque_iterator< _ITp, _IRef, _IPtr > __last, ::_Deque_iterator< _OTp, _OTp &, _OTp * > __result)
- template<bool _IsMove, typename _Tp, typename _Ref, typename _Ptr, typename _OI>
  _OI **std::__copy_move_backward_a1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last, _OI __result)
- template<bool _IsMove, typename _BI1, typename _BI2>
  constexpr _BI2 **std::__copy_move_backward_a1** (_BI1 __first, _BI1 __last, _BI2 __result)
- template<bool _IsMove, typename _II, typename _Tp>
  __gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value,::_Deque_iterator< _Tp, _Tp &, _Tp * > >::__type **std::__copy_move_backward_a1** (_II __first, _II __last, ::_Deque_iterator< _Tp, _Tp &, _Tp * > __result)
- template<bool _IsMove, typename _BI1, typename _BI2>
  constexpr _BI2 **std::__copy_move_backward_a2** (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _InputIterator, typename _Size, typename _OutputIterator>
  constexpr _OutputIterator **std::__copy_n_a** (_InputIterator __first, _Size __n, _OutputIterator __result, bool)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value,::_Deque_iterator< _CharT, _CharT &, _CharT * > >::__type **std::__copy_n_a** (istreambuf_iterator< _CharT, char_traits< _CharT > > __it, _Size __size, ::_Deque_iterator< _CharT, _CharT &, _CharT * > __result, bool __strict)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **std::__copy_n_a** (istreambuf_iterator< _CharT, char_traits< _CharT > >, _Size, _CharT *, bool)
- template<typename _InputIterator, typename _Predicate>
  constexpr iterator_traits< _InputIterator >::difference_type **std::__count_if** (_InputIterator __first, _InputIterator __last, _Predicate __pred)

- template<typename _II1, typename _II2>
  constexpr bool **std::__equal4** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1, typename _II2, typename _BinaryPredicate>
  constexpr bool **std::__equal4** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _BinaryPredicate __binary↩
  _pred)
- template<typename _II1, typename _II2>
  constexpr bool **std::__equal_aux** (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _II1, typename _II2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__equal_aux** (_II1, _II1, const ::__gnu_debug::_Safe_iterator< _II2, _Seq2, _Cat2 > &)
- template<typename _II1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **std::__equal_aux** (const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, _II2)
- template<typename _II1, typename _Seq1, typename _Cat1, typename _II2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__equal_aux** (const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II2, _Seq2, _Cat2 > &)
- template<typename _Tp, typename _Ref, typename _Ptr, typename _II>
  __gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value, bool >::__type **std::__equal_aux1** (::_Deque_iterator< _Tp, _Ref, _Ptr > __first1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __last1, _II __first2)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **std::__equal_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2)
- template<typename _II, typename _Tp, typename _Ref, typename _Ptr>
  __gnu_cxx::__enable_if< __is_any_random_access_iter< _II >::__value, bool >::__type **std::__equal_aux1** (_II __first1, _II __last1, ::_Deque_iterator< _Tp, _Ref, _Ptr > __first2)
- template<typename _II1, typename _II2>
  constexpr bool **std::__equal_aux1** (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _FIte, typename _Tp>
  constexpr void **std::__fill_a** (_FIte __first, _FIte __last, const _Tp &__value)
- template<typename _Ite, typename _Seq, typename _Cat, typename _Tp>
  constexpr void **std::__fill_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const _Tp &)
- template<typename _Ite, typename _Cont, typename _Tp>
  constexpr void **std::__fill_a1** (::__gnu_cxx::__normal_iterator< _Ite, _Cont > __first, ::__gnu_cxx::__normal_↩
  iterator< _Ite, _Cont > __last, const _Tp &__value)
- constexpr void **std::__fill_a1** (::_Bit_iterator, ::_Bit_iterator, const bool &)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void **std::__fill_a1** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _Up, typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_byte< _Up >::__value &&(__are_same< _Up, _Tp >::__value||_↩
  _memcpyable_integer< _Tp >::__width), void >::__type **std::__fill_a1** (_Up *__first, _Up *__last, const _Tp &__x)
- template<typename _Tp, typename _VTp>
  void **std::__fill_a1** (const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__first, const ::_Deque_iterator< _Tp, _Tp &, _Tp * > &__last, const _VTp &__value)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **std::__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value, std::input_iterator_tag)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **std::__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value, std::output_iterator_tag)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **std::__fill_n_a** (_OutputIterator __first, _Size __n, const _Tp &__value, std::random_access_iterator_tag)

- template<typename _Ite, typename _Seq, typename _Cat, typename _Size, typename _Tp>
  constexpr ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **std::__fill_n_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &__first, _Size __n, const _Tp &__value, std::input_iterator_tag)
- template<typename _OutputIterator, typename _Size, typename _Tp>
  constexpr _OutputIterator **std::__fill_n_a1** (_OutputIterator __first, _Size __n, const _Tp &__value)
- template<typename _Iterator, typename _Predicate>
  constexpr _Iterator **std::__find_if** (_Iterator __first, _Iterator __last, _Predicate __pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr bool **std::__is_permutation** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _BinaryPredicate __pred)
- template<typename _II1, typename _II2>
  constexpr bool **std::__lexicographical_compare_aux** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1, typename _Iter2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__lexicographical_compare_aux** (_II1, _II1, const ::__gnu_debug::_Safe_iterator< _Iter2, _Seq2, _Cat2 > &, const ::__gnu_debug::_Safe_iterator< _Iter2, _Seq2, _Cat2 > &)
- template<typename _Iter1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **std::__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, ↩ _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _Cat1 > &, _II2, _II2)
- template<typename _Iter1, typename _Seq1, typename _Cat1, typename _Iter2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, ↩ _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _Iter1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _Iter2, _Seq2, _Cat2 > &, const ::__gnu_debug::_Safe_iterator< _Iter2, _Seq2, _Cat2 > &)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **std::__lexicographical_compare_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::_Deque_iterator< _Tp2, _↩ Ref2, _Ptr2 > __last2)
- template<typename _Tp1, typename _Ref1, typename _Ptr1, typename _Tp2>
  bool **std::__lexicographical_compare_aux1** (::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __first1, ::_Deque_iterator< _Tp1, _Ref1, _Ptr1 > __last1, _Tp2 *__first2, _Tp2 *__last2)
- template<typename _II1, typename _II2>
  constexpr bool **std::__lexicographical_compare_aux1** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _Tp1, typename _Tp2, typename _Ref2, typename _Ptr2>
  bool **std::__lexicographical_compare_aux1** (_Tp1 *__first1, _Tp1 *__last1, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __first2, ::_Deque_iterator< _Tp2, _Ref2, _Ptr2 > __last2)
- template<typename _II1, typename _II2, typename _Compare>
  constexpr bool **std::__lexicographical_compare_impl** (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _↩ Compare __comp)
- template<typename _Tp>
  constexpr _Tp std::__lg (_Tp __n)
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  constexpr _ForwardIterator **std::__lower_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _Tp, typename _Up>
  constexpr int **std::__memcmp** (const _Tp *__first1, const _Up *__first2, size_t __num)
- template<typename _Tp>
  constexpr auto **std::__min_cmp** (_Tp __x, _Tp __y)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > **std::__mismatch** (_InputIterator1 __first1, _InputIterator1 ↩ __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > **std::__mismatch** (_InputIterator1 __first1, _InputIterator1 ↩ __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)

- template<typename _ForwardIterator, typename _Predicate>
  constexpr _ForwardIterator **std::__remove_if** (_ForwardIterator __first, _ForwardIterator __last, _Predicate __↩
  pred)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 **std::__search** (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _Forward↩
  Iterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)
- constexpr long long **std::__size_to_integer** (double __n)
- constexpr long long **std::__size_to_integer** (float __n)
- constexpr int **std::__size_to_integer** (int __n)
- constexpr long **std::__size_to_integer** (long __n)
- constexpr long long **std::__size_to_integer** (long double __n)
- constexpr long long **std::__size_to_integer** (long long __n)
- constexpr unsigned **std::__size_to_integer** (unsigned __n)
- constexpr unsigned long **std::__size_to_integer** (unsigned long __n)
- constexpr unsigned long long **std::__size_to_integer** (unsigned long long __n)
- template<typename _II, typename _OI>
  constexpr _OI std::copy (_II __first, _II __last, _OI __result)
- template<typename _BI1, typename _BI2>
  constexpr _BI2 std::copy_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _II1, typename _II2>
  constexpr bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2)
- template<typename _II1, typename _II2>
  constexpr bool std::equal (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _BinaryPredicate __binary_pred)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool std::equal (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _IIter2 __last2, _BinaryPredicate __↩
  binary_pred)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void std::fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _OI, typename _Size, typename _Tp>
  constexpr _OI std::fill_n (_OI __first, _Size __n, const _Tp &__value)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr bool std::is_permutation (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 _↩
  _first2)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr void std::iter_swap (_ForwardIterator1 __a, _ForwardIterator2 __b)
- template<typename _II1, typename _II2>
  constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2)
- template<typename _II1, typename _II2, typename _Compare>
  constexpr bool std::lexicographical_compare (_II1 __first1, _II1 __last1, _II2 __first2, _II2 __last2, _Compare
  __comp)
- template<typename _InputIter1, typename _InputIter2>
  constexpr auto **std::lexicographical_compare_three_way** (_InputIter1 __first1, _InputIter1 __last1, _InputIter2
  __first2, _InputIter2 __last2)
- template<typename _InputIter1, typename _InputIter2, typename _Comp>
  constexpr auto std::lexicographical_compare_three_way (_InputIter1 __first1, _InputIter1 __last1, _InputIter2 _↩
  _first2, _InputIter2 __last2, _Comp __comp) -> decltype(__comp(*__first1, *__first2))
- template<typename _ForwardIterator, typename _Tp>
  constexpr _ForwardIterator std::lower_bound (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val)
- template<typename _Tp>
  constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b)

- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::max (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _Tp>
  constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b)
- template<typename _Tp, typename _Compare>
  constexpr const _Tp & std::min (const _Tp &__a, const _Tp &__b, _Compare __comp)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  constexpr pair< _InputIterator1, _InputIterator2 > std::mismatch (_InputIterator1 __first1, _InputIterator1 __last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred)
- template<typename _II, typename _OI>
  constexpr _OI std::move (_II __first, _II __last, _OI __result)
- template<typename _BI1, typename _BI2>
  constexpr _BI2 std::move_backward (_BI1 __first, _BI1 __last, _BI2 __result)
- template<typename _ForwardIterator1, typename _ForwardIterator2, typename _BinaryPredicate>
  constexpr _ForwardIterator1 std::search (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2, _ForwardIterator2 __last2, _BinaryPredicate __predicate)
- template<typename _ForwardIterator1, typename _ForwardIterator2>
  constexpr _ForwardIterator2 std::swap_ranges (_ForwardIterator1 __first1, _ForwardIterator1 __last1, _ForwardIterator2 __first2)

7.165.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <algorithm>.

7.166 stl_bvector.h File Reference

Classes

- struct std::hash<::vector< bool, _Alloc > >
- class std::vector< bool, _Alloc >

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_ALWAYS_INLINE**

Typedefs

- typedef unsigned long **std::_Bit_type**

Enumerations

- enum { **_S_word_bit** }

Functions

- constexpr void **std::__fill_a1** (::_Bit_iterator, ::_Bit_iterator, const bool &)
- constexpr void **std::__fill_bvector** (_Bit_type *__v, unsigned int __first, unsigned int __last, bool __x) noexcept
- constexpr void **std::__fill_bvector_n** (_Bit_type *, size_t, bool) noexcept

7.166.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <vector>.

7.167 stl_construct.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Tp, typename... _Args>
  constexpr void std::_Construct (_Tp *__p, _Args &&... __args)
- template<typename _T1>
  constexpr void **std::_Construct_novalue** (_T1 *__p)
- template<typename _ForwardIterator>
  constexpr void std::_Destroy (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _Tp>
  constexpr void std::_Destroy (_Tp *__pointer)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator std::_Destroy_n (_ForwardIterator __first, _Size __count)
- template<typename _Tp, typename... _Args>
  constexpr _Tp * **std::construct_at** (_Tp *__location, _Args &&... __args) noexcept(noexcept(::new((void *) 0) _Tp(std::declval< _Args >()...)))
- template<typename _ForwardIterator>
  constexpr void **std::destroy** (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _Tp>
  constexpr void **std::destroy_at** (_Tp *__location)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator **std::destroy_n** (_ForwardIterator __first, _Size __count)
- template<typename _Tp>
  const _Tp * **std::start_lifetime_as** (const void *__p) noexcept
- template<typename _Tp>
  const volatile _Tp * **std::start_lifetime_as** (const volatile void *__p) noexcept
- template<typename _Tp>
  _Tp * **std::start_lifetime_as** (void *__p) noexcept
- template<typename _Tp>
  volatile _Tp * **std::start_lifetime_as** (volatile void *__p) noexcept
- template<typename _Tp>
  const _Tp * **std::start_lifetime_as_array** (const void *__p, size_t __n) noexcept
- template<typename _Tp>
  const volatile _Tp * **std::start_lifetime_as_array** (const volatile void *__p, size_t __n) noexcept

- template<typename _Tp>
  _Tp * **std::start_lifetime_as_array** (void *__p, size_t __n) noexcept
- template<typename _Tp>
  volatile _Tp * **std::start_lifetime_as_array** (volatile void *__p, size_t __n) noexcept

7.167.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.168 stl_deque.h File Reference

Classes

- class std::_Deque_base< _Tp, _Alloc >
- struct std::_Deque_iterator< _Tp, _Ref, _Ptr >
- class std::deque< _Tp, _Alloc >

Namespaces

- namespace std

Macros

- #define _GLIBCXX_DEQUE_BUF_SIZE

Functions

- constexpr size_t **std::__deque_buf_size** (size_t __size)
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::deque (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> deque< _ValT, _Allocator >
- template<typename _Tp, typename _Alloc>
  __detail::__synth3way_t< _Tp > std::operator<=> (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool std::operator== (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  void std::swap (deque< _Tp, _Alloc > &__x, deque< _Tp, _Alloc > &__y) noexcept(/*conditional */)

7.168.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <deque>.

7.168.2 Macro Definition Documentation

_GLIBCXX_DEQUE_BUF_SIZE

```
#define _GLIBCXX_DEQUE_BUF_SIZE
```

This function controls the size of memory nodes.

Parameters

__size	The size of an element.

Returns

The number (not byte size) of elements per node.

This function started off as a compiler kludge from SGI, but seems to be a useful wrapper around a repeated constant expression. The **512** is tunable (and no other code needs to change), but no investigation has been done since inheriting the SGI code. Touch _GLIBCXX_DEQUE_BUF_SIZE only if you know what you are doing, however: changing it breaks the binary compatibility!!

7.169 stl_function.h File Reference

Classes

- struct std::binary_function< _Arg1, _Arg2, _Result >
- class std::binary_negate< _Predicate >
- class std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >
- class std::const_mem_fun1_t< _Ret, _Tp, _Arg >
- class std::const_mem_fun_ref_t< _Ret, _Tp >
- class std::const_mem_fun_t< _Ret, _Tp >
- struct std::divides< _Tp >
- struct std::divides< void >
- struct std::equal_to< _Tp >
- struct std::equal_to< void >
- struct std::greater< _Tp >
- struct std::greater< void >
- struct std::greater_equal< _Tp >
- struct std::greater_equal< void >
- struct std::less< _Tp >
- struct std::less< void >
- struct std::less_equal< _Tp >
- struct std::less_equal< void >
- struct std::logical_and< _Tp >
- struct std::logical_and< void >
- struct std::logical_not< _Tp >
- struct std::logical_not< void >
- struct std::logical_or< _Tp >
- struct std::logical_or< void >
- class std::mem_fun1_ref_t< _Ret, _Tp, _Arg >
- class std::mem_fun1_t< _Ret, _Tp, _Arg >
- class std::mem_fun_ref_t< _Ret, _Tp >
- class std::mem_fun_t< _Ret, _Tp >
- struct std::minus< _Tp >
- struct std::minus< void >
- struct std::modulus< _Tp >
- struct std::modulus< void >
- struct std::multiplies< _Tp >
- struct std::multiplies< void >
- struct std::negate< _Tp >
- struct std::negate< void >
- struct std::not_equal_to< _Tp >
- struct std::not_equal_to< void >
- struct std::plus< _Tp >
- class std::pointer_to_binary_function< _Arg1, _Arg2, _Result >
- class std::pointer_to_unary_function< _Arg, _Result >
- struct std::unary_function< _Arg, _Result >
- class std::unary_negate< _Predicate >

Namespaces

- namespace std

Typedefs

- template<typename _Func, typename _SfinaeType>
  using **std::__has_is_transparent_t**

Functions

- template<typename _Ret, typename _Tp>
  const_mem_fun_t< _Ret, _Tp > **std::mem_fun** (_Ret(_Tp::*__f)() const)
- template<typename _Ret, typename _Tp>
  mem_fun_t< _Ret, _Tp > **std::mem_fun** (_Ret(_Tp::*__f)())
- template<typename _Ret, typename _Tp, typename _Arg>
  const_mem_fun1_t< _Ret, _Tp, _Arg > **std::mem_fun** (_Ret(_Tp::*__f)(_Arg) const)
- template<typename _Ret, typename _Tp, typename _Arg>
  mem_fun1_t< _Ret, _Tp, _Arg > **std::mem_fun** (_Ret(_Tp::*__f)(_Arg))
- template<typename _Ret, typename _Tp>
  const_mem_fun_ref_t< _Ret, _Tp > **std::mem_fun_ref** (_Ret(_Tp::*__f)() const)
- template<typename _Ret, typename _Tp>
  mem_fun_ref_t< _Ret, _Tp > **std::mem_fun_ref** (_Ret(_Tp::*__f)())
- template<typename _Ret, typename _Tp, typename _Arg>
  const_mem_fun1_ref_t< _Ret, _Tp, _Arg > **std::mem_fun_ref** (_Ret(_Tp::*__f)(_Arg) const)
- template<typename _Ret, typename _Tp, typename _Arg>
  mem_fun1_ref_t< _Ret, _Tp, _Arg > **std::mem_fun_ref** (_Ret(_Tp::*__f)(_Arg))
- template<typename _Predicate>
  constexpr unary_negate< _Predicate > std::not1 (const _Predicate &__pred)
- template<typename _Predicate>
  constexpr binary_negate< _Predicate > std::not2 (const _Predicate &__pred)
- template<typename _Arg, typename _Result>
  pointer_to_unary_function< _Arg, _Result > std::ptr_fun (_Result(*__x)(_Arg))
- template<typename _Arg1, typename _Arg2, typename _Result>
  pointer_to_binary_function< _Arg1, _Arg2, _Result > std::ptr_fun (_Result(*__x)(_Arg1, _Arg2))

Variables

- template<template< typename > class>
  constexpr bool **std::__is_std_op_template**
- template<> constexpr bool **std::__is_std_op_template**< **std::bit_and** >
- template<> constexpr bool **std::__is_std_op_template**< **std::bit_not** >
- template<> constexpr bool **std::__is_std_op_template**< **std::bit_or** >
- template<> constexpr bool **std::__is_std_op_template**< **std::bit_xor** >
- template<> constexpr bool **std::__is_std_op_template**< **std::divides** >
- template<> constexpr bool **std::__is_std_op_template**< **std::equal_to** >
- template<> constexpr bool **std::__is_std_op_template**< **std::greater** >
- template<> constexpr bool **std::__is_std_op_template**< **std::greater_equal** >
- template<> constexpr bool **std::__is_std_op_template**< **std::less** >
- template<> constexpr bool **std::__is_std_op_template**< **std::less_equal** >
- template<> constexpr bool **std::__is_std_op_template**< **std::logical_and** >
- template<> constexpr bool **std::__is_std_op_template**< **std::logical_not** >
- template<> constexpr bool **std::__is_std_op_template**< **std::logical_or** >

- template<> constexpr bool **std::__is_std_op_template< std::minus >**
- template<> constexpr bool **std::__is_std_op_template< std::modulus >**
- template<> constexpr bool **std::__is_std_op_template< std::multiplies >**
- template<> constexpr bool **std::__is_std_op_template< std::negate >**
- template<> constexpr bool **std::__is_std_op_template< std::not_equal_to >**
- template<> constexpr bool **std::__is_std_op_template< std::plus >**
- template<typename _Fn>
  constexpr bool **std::__is_std_op_wrapper**
- template<template< typename > class _Ft, typename _Tp>
  constexpr bool **std::__is_std_op_wrapper< _Ft< _Tp > >**

7.169.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <functional>.

7.170 stl_heap.h File Reference

Namespaces

- namespace std

Functions

- template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>
  constexpr void **std::__adjust_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __len, _Tp __value, _Compare __comp)
- template<typename _RandomAccessIterator, typename _Compare, typename _Distance>
  constexpr bool **std::__is_heap** (_RandomAccessIterator __first, _Compare __comp, _Distance __n)
- template<typename _RandomAccessIterator, typename _Distance>
  constexpr bool **std::__is_heap** (_RandomAccessIterator __first, _Distance __n)
- template<typename _RandomAccessIterator>
  constexpr bool **std::__is_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr bool **std::__is_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare _↩ _comp)
- template<typename _RandomAccessIterator, typename _Distance, typename _Compare>
  constexpr _Distance **std::__is_heap_until** (_RandomAccessIterator __first, _Distance __n, _Compare &__↩ comp)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **std::__make_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **std::__pop_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Random↩ AccessIterator __result, _Compare &__comp)
- template<typename _RandomAccessIterator, typename _Distance, typename _Tp, typename _Compare>
  constexpr void **std::__push_heap** (_RandomAccessIterator __first, _Distance __holeIndex, _Distance __top↩ Index, _Tp __value, _Compare &__comp)
- template<typename _RandomAccessIterator, typename _Compare>
  constexpr void **std::__sort_heap** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Compare &__comp)
- template<typename _RandomAccessIterator>
  constexpr bool std::is_heap (_RandomAccessIterator __first, _RandomAccessIterator __last)

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

7.170.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <queue>.

7.171 bits/stl_iterator.h File Reference

Classes

- class std::back_insert_iterator< _Container >
- class std::common_iterator< _It, _Sent >
- class std::counted_iterator< _It >
- class std::front_insert_iterator< _Container >
- class std::insert_iterator< _Container >
- class std::move_iterator< _Iterator >
- class std::move_sentinel< _Sent >
- class std::reverse_iterator< _Iterator >

Namespaces

- namespace __gnu_cxx
- namespace __gnu_debug
- namespace std

Macros

- #define **_GLIBCXX_MAKE_MOVE_IF_NOEXCEPT_ITERATOR**(_Iter)
- #define **_GLIBCXX_MAKE_MOVE_ITERATOR**(_Iter)

Typedefs

- template<typename _InputIterator>
  using **std::__iter_key_t**
- template<typename _InputIterator>
  using **std::__iter_to_alloc_t**
- template<typename _InputIterator>
  using **std::__iter_val_t**
- template<input_iterator _It>
  using **std::const_iterator**
- template<semiregular _Sent>
  using **std::const_sentinel**
- template<indirectly_readable _It>
  using **std::iter_const_reference_t**

Functions

- template<typename _Iterator, typename _ReturnType = __conditional_t<__move_if_noexcept_cond <typename iterator_traits<_↩ Iterator>::value_type>::value, _Iterator, move_iterator<_Iterator>>>
  constexpr _ReturnType **std::__make_move_if_noexcept_iterator** (_Iterator __i)
- template<typename _Tp, typename _ReturnType = __conditional_t<__move_if_noexcept_cond<_Tp>::value, const _Tp*, move_↩ iterator<_Tp*>>>
  constexpr _ReturnType **std::__make_move_if_noexcept_iterator** (_Tp *__i)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > **std::__make_reverse_iterator** (_Iterator __i)
- template<typename _Container>
  constexpr back_insert_iterator< _Container > std::back_inserter (_Container &__x)
- template<typename _Container>
  constexpr front_insert_iterator< _Container > std::front_inserter (_Container &__x)
- template<typename _Container>
  constexpr insert_iterator< _Container > std::inserter (_Container &__x, std::__detail::__range_iter_t< _↩ Container > __i)
- template<input_iterator _It>
  constexpr const_iterator< _It > **std::make_const_iterator** (_It __it) noexcept(is_nothrow_convertible_v< _It, const_iterator< _It > >)
- template<semiregular _Sent>
  constexpr const_sentinel< _Sent > **std::make_const_sentinel** (_Sent __s) noexcept(is_nothrow_convertible↩ _v< _Sent, const_sentinel< _Sent > >)
- template<typename _Iterator>
  constexpr move_iterator< _Iterator > **std::make_move_iterator** (_Iterator __i)
- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > std::make_reverse_iterator (_Iterator __i)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() != __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator!=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Iterator>
  requires requires { { __x.base() + __n } -> same_as<_Iterator>; }
  constexpr move_iterator< _Iterator > **std::operator+** (typename move_iterator< _Iterator >::difference_type ↩ __n, const move_iterator< _Iterator > &__x)

- template<typename _Iterator>
  constexpr reverse_iterator< _Iterator > **std::operator+** (typename reverse_iterator< _Iterator >::difference_type __n, const reverse_iterator< _Iterator > &__x)
- template<typename _IteratorL, typename _IteratorR>
  constexpr auto **std::operator-** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y) -> decltype(__x.base() - __y.base())
- template<typename _IteratorL, typename _IteratorR>
  constexpr auto **std::operator-** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y) -> decltype(__y.base() - __x.base())
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator<** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() > __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() >= __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator<=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<three_way_comparable _Iterator>
  constexpr compare_three_way_result_t< _Iterator > **std::operator<=>** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
  constexpr compare_three_way_result_t< _IteratorL, _IteratorR > **std::operator<=>** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<three_way_comparable _Iterator>
  constexpr compare_three_way_result_t< _Iterator, _Iterator > **std::operator<=>** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL, three_way_comparable_with< _IteratorL > _IteratorR>
  constexpr compare_three_way_result_t< _IteratorL, _IteratorR > **std::operator<=>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _Iterator>
  constexpr bool **std::operator==** (const move_iterator< _Iterator > &__x, const move_iterator< _Iterator > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator==** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _Iterator>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator==** (const reverse_iterator< _Iterator > &__x, const reverse_iterator< _Iterator > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() == __y.base() } -> convertible_to<bool>; }
  constexpr bool std::operator== (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __y.base() < __x.base() } -> convertible_to<bool>; }

constexpr bool **std::operator>** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)

- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() < __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>=** (const move_iterator< _IteratorL > &__x, const move_iterator< _IteratorR > &__y)
- template<typename _IteratorL, typename _IteratorR>
  requires requires { { __x.base() <= __y.base() } -> convertible_to<bool>; }
  constexpr bool **std::operator>=** (const reverse_iterator< _IteratorL > &__x, const reverse_iterator< _IteratorR > &__y)

Variables

- template<typename _Iterator1, typename _Iterator2>
  constexpr bool **std::disable_sized_sentinel_for**< **move_iterator**< **_Iterator1** >, **move_iterator**< **_Iterator2** > >
- template<typename _Iterator1, typename _Iterator2>
  constexpr bool **std::disable_sized_sentinel_for**< **reverse_iterator**< **_Iterator1** >, **reverse_iterator**< **_Iterator2** > >

7.171.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.
This file implements reverse_iterator, back_insert_iterator, front_insert_iterator, insert_iterator, __normal_iterator, and their supporting functions and overloaded operators.

7.172 debug/stl_iterator.h File Reference

Namespaces

- namespace __gnu_debug

Functions

- template<typename _Iterator>
  constexpr auto **__gnu_debug::__base** (const std::move_iterator< _Iterator > &__it) -> decltype(std::make_move_iterator(__base(__it.base())))
- template<typename _Iterator, typename _Sequence>
  constexpr std::reverse_iterator< _Iterator > **__gnu_debug::__base** (const std::reverse_iterator< _Safe_iterator< _Iterator, _Sequence, std::random_access_iterator_tag > > &__it)
- template<typename _Iterator, typename _Size>
  constexpr bool **__gnu_debug::__can_advance** (const std::move_iterator< _Iterator > &__it, _Size __n)
- template<typename _Iterator, typename _Diff>
  constexpr bool **__gnu_debug::__can_advance** (const std::move_iterator< _Iterator > &__it, const std::pair< _Diff, _Distance_precision > &__dist, int __way)
- template<typename _Iterator, typename _Size>
  constexpr bool **__gnu_debug::__can_advance** (const std::reverse_iterator< _Iterator > &__it, _Size __n)
- template<typename _Iterator, typename _Diff>
  constexpr bool **__gnu_debug::__can_advance** (const std::reverse_iterator< _Iterator > &__it, const std::pair< _Diff, _Distance_precision > &__dist, int __way)

- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__gnu_debug::__get_distance** (const std::move_iterator< ↩
  _Iterator > &__first, const std::move_iterator< _Iterator > &__last)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__gnu_debug::__get_distance** (const std::reverse_iterator< _Iterator > &__first, const std::reverse_iterator< _Iterator > &__last)
- template<typename _Iterator>
  constexpr auto **__gnu_debug::__unsafe** (const std::move_iterator< _Iterator > &__it) -> decltype(std::make↩
  _move_iterator(__unsafe(__it.base())))
- template<typename _Iterator>
  constexpr auto **__gnu_debug::__unsafe** (const std::reverse_iterator< _Iterator > &__it) -> decltype(std::__↩
  make_reverse_iterator(__unsafe(__it.base())))
- template<typename _Iterator>
  constexpr bool **__gnu_debug::__valid_range** (const std::move_iterator< _Iterator > &__first, const std::move_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)
- template<typename _Iterator>
  constexpr bool **__gnu_debug::__valid_range** (const std::reverse_iterator< _Iterator > &__first, const std::reverse_iterator< _Iterator > &__last, typename _Distance_traits< _Iterator >::__type &__dist)

7.172.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.173 stl_iterator_base_funcs.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Tp>
  ptrdiff_t **std::__distance** (::_List_const_iterator< _Tp >, ::_List_const_iterator< _Tp >, input_iterator_tag)
- template<typename _Tp>
  ptrdiff_t **std::__distance** (::_List_iterator< _Tp >, ::_List_iterator< _Tp >, input_iterator_tag)
- template<typename _InputIterator>
  constexpr iterator_traits< _InputIterator >::difference_type **std::__distance** (_InputIterator __first, _InputIterator __last, input_iterator_tag)
- template<typename _OutputIterator>
  void **std::__distance** (_OutputIterator, _OutputIterator, output_iterator_tag)=delete
- template<typename _RandomAccessIterator>
  constexpr iterator_traits< _RandomAccessIterator >::difference_type **std::__distance** (_RandomAccessIterator __first, _RandomAccessIterator __last, random_access_iterator_tag)
- template<typename _InputIterator, typename _Distance>
  constexpr void std::advance (_InputIterator &__i, _Distance __n)
- template<typename _InputIterator>
  constexpr iterator_traits< _InputIterator >::difference_type std::distance (_InputIterator __first, _InputIterator _↩
  _last)
- template<typename _InputIterator>
  constexpr _InputIterator **std::next** (_InputIterator __x, typename iterator_traits< _InputIterator >::difference_type __n=1)
- template<typename _BidirectionalIterator>
  constexpr _BidirectionalIterator **std::prev** (_BidirectionalIterator __x, typename iterator_traits< _Bidirectional↩
  Iterator >::difference_type __n=1)

7.173.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.
This file contains all of the general iterator-related utility functions, such as distance() and advance().

7.174 stl_iterator_base_types.h File Reference

Classes

- struct std::bidirectional_iterator_tag
- struct std::contiguous_iterator_tag
- struct std::forward_iterator_tag
- struct std::input_iterator_tag
- struct std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >
- struct std::iterator_traits< _Iterator >
- struct std::iterator_traits< _Tp * >
- struct std::output_iterator_tag
- struct std::random_access_iterator_tag

Namespaces

- namespace std

Macros

- #define **_GLIBCXX26_ALGO_DEF_VAL_T**(_Iterator)
- #define **_GLIBCXX26_DEF_VAL_T**(T)

7.174.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.
This file contains all of the general iterator-related utility types, such as iterator_traits and struct iterator.

7.175 stl_list.h File Reference

Classes

- class std::_List_base< _Tp, _Alloc >
- struct std::_List_const_iterator< _Tp >
- struct std::_List_iterator< _Tp >
- struct std::_List_node< _Tp >
- struct std::__detail::_List_node_base
- struct std::__detail::_List_node_header
- class std::list< _Tp, _Alloc >

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_LIST_REMOVE_RETURN_TYPE_TAG**
- #define **_GLIBCXX_USE_ALLOC_PTR_FOR_LIST**

Functions

- template<typename _Tp>
  ptrdiff_t **std::__distance** (::_List_const_iterator< _Tp >, ::_List_const_iterator< _Tp >, input_iterator_tag)
- template<typename _Tp>
  ptrdiff_t **std::__distance** (::_List_iterator< _Tp >, ::_List_iterator< _Tp >, input_iterator_tag)
- template<bool _Const, typename _Ptr>
  ptrdiff_t **std::__distance** (__list::_Iterator< _Const, _Ptr > __first, __list::_Iterator< _Const, _Ptr > __last, input_iterator_tag __tag)
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> list< _ValT, _Allocator >
- template<typename _Tp, typename _Alloc>
  __detail::__synth3way_t< _Tp > std::operator<=> (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool std::operator== (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  void std::swap (list< _Tp, _Alloc > &__x, list< _Tp, _Alloc > &__y) noexcept(/*conditional */)

7.175.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <list>.

7.176 stl_map.h File Reference

Classes

- class std::map< _Key, _Tp, _Compare, _Alloc >

Namespaces

- namespace std

Functions

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::map (_InputIterator, _InputIterator, _Allocator) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter↩ _to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::map (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> map< __iter_key↩ _t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> map< _Key, _Tp, less< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::map (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> map< _↩ Key, _Tp, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< pair< const _Key, _Tp > > std::operator<=> (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)

- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator== (const map< _Key, _Tp, _Compare, _Alloc > &__x, const map< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  void std::swap (map< _Key, _Tp, _Compare, _Alloc > &__x, map< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)

7.176.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <map>.

7.177 stl_multimap.h File Reference

Classes

- class std::multimap< _Key, _Tp, _Compare, _Alloc >

Namespaces

- namespace std

Functions

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::multimap (_InputIterator, _InputIterator, _Allocator) -> multimap< __iter_key_t< _InputIterator >, __iter↩ _val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter↩ _to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::multimap (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< ↩ __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> multimap< _Key, _Tp, less< _Key >, _↩ Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::multimap (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< _Key, _Tp, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator!= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator< (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator<= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator== (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator> (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _↩ Compare, _Alloc > &__y)

- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  bool std::operator>= (const multimap< _Key, _Tp, _Compare, _Alloc > &__x, const multimap< _Key, _Tp, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  void std::swap (multimap< _Key, _Tp, _Compare, _Alloc > &__x, multimap< _Key, _Tp, _Compare, _Alloc > &__y) noexcept(/*conditional */)

7.177.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <map>.

7.178 stl_multiset.h File Reference

Classes

- class std::multiset< _Key, _Compare, _Alloc >

Namespaces

- namespace std

Functions

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::multiset (_InputIterator, _InputIterator, _Allocator) -> multiset< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::multiset (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::multiset (initializer_list< _Key >, _Allocator) -> multiset< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::multiset (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< _Key, _Compare, _Allocator >
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator!= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator< (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator<= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator== (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator> (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)

- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator>= (const multiset< _Key, _Compare, _Alloc > &__x, const multiset< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  void std::swap (multiset< _Key, _Compare, _Alloc > &__x, multiset< _Key, _Compare, _Alloc > &__y) noexcept(/*conditional */)

7.178.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <set>.

7.179 stl_numeric.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_MOVE_IF_20**(_E)

Functions

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

7.179.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <numeric>.

7.180 stl_pair.h File Reference

Classes

- struct std::pair< _T1, _T2 >
- struct std::piecewise_construct_t
- struct std::tuple_element< 0, pair< _Tp1, _Tp2 > >
- struct std::tuple_element< 1, pair< _Tp1, _Tp2 > >
- struct std::tuple_size< pair< _Tp1, _Tp2 > >

Namespaces

- namespace std

Functions

- template<typename _T1, typename _T2>
  constexpr pair< typename __decay_and_strip< _T1 >::__type, typename __decay_and_strip< _T2 >::__type > std::make_pair (_T1 &&__x, _T2 &&__y)

- template<typename _Tp, typename _Up>
  constexpr const _Tp && std::get (const pair< _Tp, _Up > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp & std::get (const pair< _Tp, _Up > &__p) noexcept
- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (const pair< _Tp1, _Tp2 > &&__in) noexcept
- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr const tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (const pair< _Tp1, _Tp2 > &__in) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp && std::get (const pair< _Up, _Tp > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr const _Tp & std::get (const pair< _Up, _Tp > &__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp && std::get (pair< _Tp, _Up > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp & std::get (pair< _Tp, _Up > &__p) noexcept
- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type && std::get (pair< _Tp1, _Tp2 > &&__in) noexcept
- template<size_t _Int, class _Tp1, class _Tp2>
  constexpr tuple_element< _Int, pair< _Tp1, _Tp2 > >::type & std::get (pair< _Tp1, _Tp2 > &__in) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp && std::get (pair< _Up, _Tp > &&__p) noexcept
- template<typename _Tp, typename _Up>
  constexpr _Tp & std::get (pair< _Up, _Tp > &__p) noexcept

Variables

- template<typename _Tp>
  constexpr bool **std::__is_pair**
- template<typename _Tp, typename _Up>
  constexpr bool **std::__is_pair**< **pair**< **_Tp, _Up** > >
- constexpr piecewise_construct_t std::piecewise_construct

- template<typename _Tp1, typename _Tp2>
  constexpr size_t **std::tuple_size_v**< **const pair**< **_Tp1, _Tp2** > >
- template<typename _Tp1, typename _Tp2>
  constexpr size_t **std::tuple_size_v**< **pair**< **_Tp1, _Tp2** > >

7.180.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.

7.181 stl_queue.h File Reference

Classes

- class std::priority_queue< _Tp, _Sequence, _Compare >
- class std::queue< _Tp, _Sequence >

Namespaces

- namespace std

Functions

- template<typename _Tp, typename _Seq>
  bool std::operator!= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator< (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator<= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp, three_way_comparable _Seq>
  compare_three_way_result_t< _Seq > **std::operator<=>** (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator== (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator> (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator>= (const queue< _Tp, _Seq > &__x, const queue< _Tp, _Seq > &__y)
- template<typename _Compare, typename _Container, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>
  std::priority_queue (_Compare, _Container) -> priority_queue< typename _Container::value_type, _Container, _Compare >
- template<typename _Compare, typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>
  std::priority_queue (_Compare, _Container, _Allocator) -> priority_queue< typename _Container::value_type, _Container, _Compare >
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Compare = less<_ValT>, typename _Container = vector<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireNotAllocator<_Container>>
  std::priority_queue (_InputIterator, _InputIterator, _Compare=_Compare(), _Container=_Container()) -> priority_queue< _ValT, _Container, _Compare >
- template<typename _Container, typename = _RequireNotAllocator<_Container>>
  std::queue (_Container) -> queue< typename _Container::value_type, _Container >

- template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>
  std::queue (_Container, _Allocator) -> queue< typename _Container::value_type, _Container >
- template<typename _Tp, typename _Sequence, typename _Compare>
  constexpr enable_if< __and_< __is_swappable< _Sequence >, __is_swappable< _Compare > >::value >::type **std::swap** (priority_queue< _Tp, _Sequence, _Compare > &__x, priority_queue< _Tp, _Sequence, _Compare > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Tp, typename _Seq>
  enable_if< __is_swappable< _Seq >::value >::type **std::swap** (queue< _Tp, _Seq > &__x, queue< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))

7.181.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <queue>.

7.182 stl_raw_storage_iter.h File Reference

Classes

- class std::raw_storage_iterator< _OutputIterator, _Tp >

Namespaces

- namespace std

7.182.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.183 stl_relops.h File Reference

Namespaces

- namespace std
- namespace std::rel_ops

Functions

- template<class _Tp>
  bool std::rel_ops::operator!= (const _Tp &__x, const _Tp &__y)
- template<class _Tp>
  bool std::rel_ops::operator<= (const _Tp &__x, const _Tp &__y)
- template<class _Tp>
  bool std::rel_ops::operator> (const _Tp &__x, const _Tp &__y)
- template<class _Tp>
  bool std::rel_ops::operator>= (const _Tp &__x, const _Tp &__y)

7.183.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.
This file is only included by <utility>, which is required by the standard to define namespace rel_ops and its contents.

7.184 stl_set.h File Reference

Classes

- class std::set< _Key, _Compare, _Alloc >

Namespaces

- namespace std

Functions

- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator!= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator< (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator<= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator== (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator> (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _Key, typename _Compare, typename _Alloc>
  bool std::operator>= (const set< _Key, _Compare, _Alloc > &__x, const set< _Key, _Compare, _Alloc > &__y)
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _↩ Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::set (initializer_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNot↩ Allocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::set (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _Compare, _Allocator >
- template<typename _Key, typename _Compare, typename _Alloc>
  void std::swap (set< _Key, _Compare, _Alloc > &__x, set< _Key, _Compare, _Alloc > &__y) noexcept(/↩ *conditional */)

7.184.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <set>.

7.185 stl_stack.h File Reference

Classes

- class std::stack< _Tp, _Sequence >

Namespaces

- namespace std

Functions

- template<typename _Tp, typename _Seq>
  bool std::operator!= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator< (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator<= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp, three_way_comparable _Seq>
  compare_three_way_result_t< _Seq > **std::operator<=>** (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator== (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator> (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Tp, typename _Seq>
  bool std::operator>= (const stack< _Tp, _Seq > &__x, const stack< _Tp, _Seq > &__y)
- template<typename _Container, typename = _RequireNotAllocator<_Container>>
  std::stack (_Container) -> stack< typename _Container::value_type, _Container >
- template<typename _Container, typename _Allocator, typename = _RequireNotAllocator<_Container>>
  std::stack (_Container, _Allocator) -> stack< typename _Container::value_type, _Container >
- template<typename _Tp, typename _Seq>
  enable_if< __is_swappable< _Seq >::value >::type **std::swap** (stack< _Tp, _Seq > &__x, stack< _Tp, _Seq > &__y) noexcept(noexcept(__x.swap(__y)))

7.185.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <stack>.

7.186 stl_tempbuf.h File Reference

Classes

- class std::_Temporary_buffer< _ForwardIterator, _Tp >

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_OPERATOR_DELETE**
- #define **_GLIBCXX_OPERATOR_NEW**
- #define **_GLIBCXX_SIZED_DEALLOC**(T, p, n)

Functions

- template<typename _Tp>
  _Tp * **std::__detail::__get_temporary_buffer** (ptrdiff_t __len) noexcept
- template<typename _Tp>
  void **std::__detail::__return_temporary_buffer** (_Tp *__p, size_t __len)
- template<typename _Tp, typename _ForwardIterator>
  void **std::__uninitialized_construct_buf** (_Tp *__first, _Tp *__last, _ForwardIterator __seed)
- template<typename _Tp>
  pair< _Tp *, ptrdiff_t > std::get_temporary_buffer (ptrdiff_t __len) noexcept
- template<typename _Tp>
  void std::return_temporary_buffer (_Tp *__p)

7.186.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.187 stl_tree.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_USE_ALLOC_PTR_FOR_RB_TREE**

Enumerations

- enum **_Rb_tree_color** { **_S_red** , **_S_black** }

Functions

- unsigned int **std::_Rb_tree_black_count** (const _Rb_tree_node_base *__node, const _Rb_tree_node_base *__root) throw ()
- _Rb_tree_node_base * **std::_Rb_tree_decrement** (_Rb_tree_node_base *__x) throw ()
- _Rb_tree_node_base * **std::_Rb_tree_increment** (_Rb_tree_node_base *__x) throw ()
- void **std::_Rb_tree_insert_and_rebalance** (const bool __insert_left, _Rb_tree_node_base *__x, _Rb_tree_node_base *__p, _Rb_tree_node_base &__header) throw ()
- _Rb_tree_node_base * **std::_Rb_tree_rebalance_for_erase** (_Rb_tree_node_base *const __z, _Rb_tree_node_base &__header) throw ()
- template<typename _Key, typename _Val, typename _KeyOfValue, typename _Compare, typename _Alloc>
  void **std::swap** (_Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__x, _Rb_tree< _Key, _Val, _KeyOfValue, _Compare, _Alloc > &__y)

7.187.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <map> or <set>.

7.188 stl_uninitialized.h File Reference

Namespaces

- namespace std

Functions

- template<typename _InputIterator, typename _ForwardIterator>
  constexpr _ForwardIterator std::uninitialized_copy (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator>
  constexpr _ForwardIterator std::uninitialized_copy_n (_InputIterator __first, _Size __n, _ForwardIterator __result)
- template<typename _ForwardIterator>
  constexpr void std::uninitialized_default_construct (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator std::uninitialized_default_construct_n (_ForwardIterator __first, _Size __count)
- template<typename _ForwardIterator, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>
  constexpr void std::uninitialized_fill (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__x)
- template<typename _ForwardIterator, typename _Size, typename _Tp = typename iterator_traits<_ForwardIterator>::value_type>
  constexpr _ForwardIterator std::uninitialized_fill_n (_ForwardIterator __first, _Size __n, const _Tp &__x)
- template<typename _InputIterator, typename _ForwardIterator>
  constexpr _ForwardIterator std::uninitialized_move (_InputIterator __first, _InputIterator __last, _ForwardIterator __result)
- template<typename _InputIterator, typename _Size, typename _ForwardIterator>
  constexpr pair< _InputIterator, _ForwardIterator > std::uninitialized_move_n (_InputIterator __first, _Size __count, _ForwardIterator __result)
- template<typename _ForwardIterator>
  constexpr void std::uninitialized_value_construct (_ForwardIterator __first, _ForwardIterator __last)
- template<typename _ForwardIterator, typename _Size>
  constexpr _ForwardIterator std::uninitialized_value_construct_n (_ForwardIterator __first, _Size __count)

7.188.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.189 stl_vector.h File Reference

Classes

- struct std::_Vector_base< _Tp, _Alloc >
- class std::vector< _Tp, _Alloc >

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_ASAN_ANNOTATE_BEFORE_DEALLOC**
- #define **_GLIBCXX_ASAN_ANNOTATE_GREW**(n)
- #define **_GLIBCXX_ASAN_ANNOTATE_GROW**(n)
- #define **_GLIBCXX_ASAN_ANNOTATE_REINIT**
- #define **_GLIBCXX_ASAN_ANNOTATE_SHRINK**(n)

Functions

- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > std::operator<=> (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  constexpr bool std::operator== (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  constexpr void std::swap (vector< _Tp, _Alloc > &__x, vector< _Tp, _Alloc > &__y) noexcept(/*conditional */)
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::vector (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> vector< _ValT, _Allocator >

7.189.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <vector>.

7.190 stream_iterator.h File Reference

Classes

- class std::istream_iterator< _Tp, _CharT, _Traits, _Dist >
- class std::ostream_iterator< _Tp, _CharT, _Traits >

Namespaces

- namespace std

7.190.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

7.191 streambuf.tcc File Reference

Namespaces

- namespace std

Functions

- template<typename _CharT, typename _Traits>
  streamsize **std::__copy_streambufs** (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< _CharT, _Traits > *__sbout)
- template<typename _CharT, typename _Traits>
  streamsize **std::__copy_streambufs_eof** (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< ↩ _CharT, _Traits > *__sbout, bool &__ineof)

7.191.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <streambuf>.

7.192 streambuf_iterator.h File Reference

Classes

- class std::istreambuf_iterator< _CharT, _Traits >
- class std::ostreambuf_iterator< _CharT, _Traits >

Namespaces

- namespace std

Functions

- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::__copy_move_a2** (_CharT *__first, _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::__copy_move_a2** (const _CharT *__first, const _CharT *__last, ostreambuf_iterator< _CharT > __result)
- template<bool _IsMove, typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **std::__copy_move_a2** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, _CharT *__result)
- template<typename _CharT, typename _Size>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, _CharT * >::__type **std::__copy_n_a** (istreambuf_iterator< _CharT > __it, _Size __n, _CharT *__result, bool __strict)
- template<typename _CharT, typename _Distance>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, void >::__type **std::advance** (istreambuf_iterator< _CharT > &__i, _Distance __n)
- template<typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, ostreambuf_iterator< _CharT > >::__type **std::copy** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, ostreambuf_iterator< _CharT > __result)
- template<typename _CharT>
  __gnu_cxx::__enable_if< __is_char< _CharT >::__value, istreambuf_iterator< _CharT > >::__type **std::find** (istreambuf_iterator< _CharT > __first, istreambuf_iterator< _CharT > __last, const _CharT &__val)
- template<typename _CharT, typename _Traits>
  bool **std::operator==** (const istreambuf_iterator< _CharT, _Traits > &__a, const istreambuf_iterator< _CharT, _Traits > &__b)

7.192.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iterator>.

7.193 bits/string_view.tcc File Reference

Namespaces

- namespace std

7.193.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string_view>.

7.194 experimental/bits/string_view.tcc File Reference

Namespaces

- namespace std
- namespace std::experimental

7.194.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <experimental/string_view>.

7.195 stringfwd.h File Reference

Namespaces

- namespace std

Typedefs

- typedef basic_string< char > std::string
- typedef basic_string< char16_t > std::u16string
- typedef basic_string< char32_t > std::u32string
- typedef basic_string< wchar_t > std::wstring

7.195.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <string>.

7.196 text_encoding-data.h File Reference

Macros

- #define **_GLIBCXX_TEXT_ENCODING_UTF8_OFFSET**

7.196.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <text_encoding>.

7.197 this_thread_sleep.h File Reference

Namespaces

- namespace std
- namespace std::this_thread

Functions

- template<typename _Rep, typename _Period>
  void std::this_thread::sleep_for (const chrono::duration< _Rep, _Period > &__rtime)
- template<typename _Clock, typename _Duration>
  void std::this_thread::sleep_until (const chrono::time_point< _Clock, _Duration > &__atime)

7.197.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <thread>.

7.198 unicode-data.h File Reference

Enumerations

- enum class **_Gcb_property** {
  _Gcb_Other , **_Gcb_Control** , **_Gcb_LF** , **_Gcb_CR** ,
  _Gcb_Extend , **_Gcb_Prepend** , **_Gcb_SpacingMark** , **_Gcb_L** ,
  _Gcb_V , **_Gcb_T** , **_Gcb_ZWJ** , **_Gcb_LV** ,
  _Gcb_LVT , **_Gcb_Regional_Indicator** }
- enum class **_InCB** { **_Consonant** , **_Extend** }

Variables

- constexpr uint32_t **__escape_edges** [ ]
- constexpr uint32_t **__gcb_edges** [ ]
- constexpr int **__gcb_shift_bits**
- constexpr uint32_t **__incb_edges** [ ]
- constexpr char32_t **__incb_linkers** [ ]
- constexpr char32_t **__width_edges** [ ]
- constexpr char32_t **__xpicto_edges** [ ]

7.198.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <format>.

7.199 unicode.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_GET_UNICODE_DATA**

Typedefs

- template<typename _View>
  using **std::__unicode::_Utf16_view**
- template<typename _View>
  using **std::__unicode::_Utf32_view**
- template<typename _View>
  using **std::__unicode::_Utf8_view**

Enumerations

- enum class **_Gcb_property** {
  _Gcb_Other , **_Gcb_Control** , **_Gcb_LF** , **_Gcb_CR** ,
  _Gcb_Extend , **_Gcb_Prepend** , **_Gcb_SpacingMark** , **_Gcb_L** ,
  _Gcb_V , **_Gcb_T** , **_Gcb_ZWJ** , **_Gcb_LV** ,
  _Gcb_LVT , **_Gcb_Regional_Indicator** }
- enum class **_InCB** { **_Consonant** , **_Extend** }

Functions

- constexpr bool **std::__unicode::__charset_alias_match** (string_view __a, string_view __b)
- template<typename _CharT>
  constexpr size_t **std::__unicode::__field_width** (basic_string_view< _CharT > __s)
- constexpr int **std::__unicode::__field_width** (char32_t __c) noexcept
- constexpr _Gcb_property **std::__unicode::__grapheme_cluster_break_property** (char32_t __c) noexcept
- constexpr _InCB **std::__unicode::__incb_property** (char32_t __c) noexcept
- constexpr bool **std::__unicode::__is_extended_pictographic** (char32_t __c)
- constexpr bool **std::__unicode::__is_incb_linker** (char32_t __c) noexcept
- constexpr bool **std::__unicode::__is_scalar_value** (char32_t __c)
- template<typename _CharT>
  constexpr bool **std::__unicode::__is_single_code_unit** (char32_t __c)
- consteval bool **std::__unicode::__literal_encoding_is_extended_ascii** ()
- template<typename _CharT>
  consteval bool **std::__unicode::__literal_encoding_is_unicode** ()
- consteval bool **std::__unicode::__literal_encoding_is_utf8** ()
- constexpr bool **std::__unicode::__should_escape_category** (char32_t __c) noexcept
- template<typename _CharT>
  constexpr size_t **std::__unicode::__truncate** (basic_string_view< _CharT > &__s, size_t __max)

Variables

- constexpr uint32_t **std::__unicode::__escape_edges** [ ]
- constexpr uint32_t **std::__unicode::__gcb_edges** [ ]
- constexpr int **std::__unicode::__gcb_shift_bits**
- constexpr uint32_t **std::__unicode::__incb_edges** [ ]
- constexpr char32_t **std::__unicode::__incb_linkers** [ ]
- constexpr char32_t **std::__unicode::__width_edges** [ ]
- constexpr char32_t **std::__unicode::__xpicto_edges** [ ]
- template<typename _Range>
  constexpr bool **std::ranges::enable_borrowed_range**< **std::__unicode::_Grapheme_cluster_view**< _↩
  Range > >
- template<typename _To, typename _Range>
  constexpr bool **std::ranges::enable_borrowed_range**< **std::__unicode::_Utf_view**< **_To, _Range** > >

7.199.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <format>.

7.200 uniform_int_dist.h File Reference

Classes

- struct std::uniform_int_distribution< _IntType >::param_type
- class std::uniform_int_distribution< _IntType >

Namespaces

- namespace std

Concepts

- concept std::uniform_random_bit_generator

7.200.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <random>.

7.201 unique_lock.h File Reference

Classes

- class std::unique_lock< _Mutex >

Namespaces

- namespace std

7.201.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <mutex>.

7.202 unique_ptr.h File Reference

Classes

- struct std::default_delete< _Tp >
- struct std::default_delete< _Tp[ ]>
- struct std::hash< unique_ptr< _Tp, _Dp > >
- class std::unique_ptr< _Tp, _Dp >
- class std::unique_ptr< _Tp[ ], _Dp >

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator< (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)

- template<typename _Tp, typename _Dp>
  constexpr bool std::operator< (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator<= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator<= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer, typename unique_ptr< _Up, _Ep >::pointer > **std::operator<=>** (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr compare_three_way_result_t< typename unique_ptr< _Tp, _Dp >::pointer > **std::operator<=>** (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator== (const unique_ptr< _Tp, _Dp > &__x, nullptr_t) noexcept
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator> (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator> (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Dp, typename _Up, typename _Ep>
  constexpr bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, const unique_ptr< _Up, _Ep > &__y)
- template<typename _Tp, typename _Dp>
  constexpr bool std::operator>= (const unique_ptr< _Tp, _Dp > &__x, nullptr_t)
- template<typename _Tp, typename _Dp>
  bool std::operator>= (nullptr_t, const unique_ptr< _Tp, _Dp > &__x)
- template<typename _Tp, typename _Dp>
  enable_if<!__is_swappable< _Dp >::value >::type **std::swap** (unique_ptr< _Tp, _Dp > &, unique_ptr< _Tp, _Dp > &)=delete

7.202.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.203 unordered_map.h File Reference

Classes

- class std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
- class std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

Namespaces

- namespace std

Typedefs

- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __umap_traits<__cache_default<_Key, _Hash>::value>>
  using **std::__umap_hashtable**
- template<bool _Cache>
  using std::__umap_traits
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = std::equal_to<_Key>, typename _Alloc = std::allocator<std::pair<const _Key, _Tp> >, typename _Tr = __ummap_traits<__cache_default<_Key, _Hash>::value>>
  using **std::__ummap_hashtable**
- template<bool _Cache>
  using std::__ummap_traits

Functions

- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  bool **std::operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  bool **std::operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  void **std::swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<class _Key, class _Tp, class _Hash, class _Pred, class _Alloc>
  void **std::swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (_InputIterator, _InputIterator, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >

- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< _Key, _Tp, _Hash, _Pred, _↩ Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::unordered_multimap (_InputIterator, _InputIterator, _Allocator) -> unordered_multimap< __iter_key↩ _t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = ↩ _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Hash, ↩ _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_↩ t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_↩ Allocator>>
  std::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type={}, _↩ Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< _Key, _Tp, _Hash, _Pred, _Allocator >

7.203.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <unordered_map>.

7.204 unordered_set.h File Reference

Classes

- class std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
- class std::unordered_set< _Value, _Hash, _Pred, _Alloc >

Namespaces

- namespace std

Typedefs

- template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::↩ allocator<_Value>, typename _Tr = __umset_traits<__cache_default<_Value, _Hash>::value>>
  using **std::__umset_hashtable**
- template<bool _Cache>
  using std::__umset_traits
- template<typename _Value, typename _Hash = hash<_Value>, typename _Pred = std::equal_to<_Value>, typename _Alloc = std::↩ allocator<_Value>, typename _Tr = __uset_traits<__cache_default<_Value, _Hash>::value>>
  using **std::__uset_hashtable**
- template<bool _Cache>
  using std::__uset_traits

Functions

- template<class _Value, class _Hash, class _Pred, class _Alloc>
  bool **std::operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_↩ multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  bool **std::operator==** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _↩ Value, _Hash, _Pred, _Alloc > &__y)
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  void **std::swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _↩ Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<class _Value, class _Hash, class _Pred, class _Alloc>
  void **std::swap** (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::unordered_multiset (_InputIterator, _InputIterator, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = ↩ _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Hash, _↩ Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >

- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_Input↩ Iterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type={}, _Hash=_↩ Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< typename iterator_traits< _Input↩ Iterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_multiset (initializer_list< _Tp >, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _↩ RequireAllocator<_Allocator>>
  std::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_↩ Allocator>>
  std::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type={}, _Hash=_Hash(), ↩ _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::unordered_set (_InputIterator, _InputIterator, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< type-name iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Allocator) -> unordered↩ _set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_Input↩ Iterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type={}, _Hash=_Hash(), ↩ _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (initializer_list< _Tp >, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< _Tp, _Hash, equal_to< _Tp >, _Allocator >

- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< _Tp, _Hash, _Pred, _Allocator >

7.204.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <unordered_set>.

7.205 uses_allocator_args.h File Reference

Namespaces

- namespace std

Functions

- template<typename _Tp, typename _Alloc, typename... _Args>
  constexpr _Tp **std::make_obj_using_allocator** (const _Alloc &__a, _Args &&... __args)
- template<typename _Tp, typename _Alloc, typename... _Args>
  constexpr _Tp * **std::uninitialized_construct_using_allocator** (_Tp *__p, const _Alloc &__a, _Args &&... _args)
- template<_Std_pair _Tp, typename _Alloc>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, _Up &&, _Vp &&) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, const pair< _Up, _Vp > &&) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, const pair< _Up, _Vp > &) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, pair< _Up, _Vp > &&) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Up, typename _Vp>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &, pair< _Up, _Vp > &) noexcept
- template<typename _Tp, typename _Alloc, typename... _Args>
  requires (! _Std_pair<_Tp>)
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &__a, _Args &&... __args) noexcept
- template<_Std_pair _Tp, typename _Alloc, typename _Tuple1, typename _Tuple2>
  constexpr auto **std::uses_allocator_construction_args** (const _Alloc &__a, piecewise_construct_t, _Tuple1 &&__x, _Tuple2 &&__y) noexcept

7.205.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.206 utility.h File Reference

Classes

- struct std::integer_sequence< _Tp, _Idx >

Namespaces

- namespace std

Typedefs

- template<typename _Tp, typename _Up = typename remove_cv<_Tp>::type, typename = typename enable_if<is_same<_Tp, _Up>::↩ value>::type, size_t = tuple_size<_Tp>::value>
  using **std::__enable_if_has_tuple_size**
- template<size_t __i, typename _Tp>
  using **std::__tuple_element_t**
- template<size_t... _Idx>
  using std::index_sequence
- template<typename... _Types>
  using std::index_sequence_for
- template<size_t _Num>
  using std::make_index_sequence
- template<typename _Tp, _Tp _Num>
  using std::make_integer_sequence
- template<size_t __i, typename _Tp>
  using **std::tuple_element_t**

Functions

- template<typename _Tp, typename... _Types>
  constexpr size_t **std::__find_uniq_type_in_pack** ()

Variables

- template<typename>
  constexpr bool **std::__is_constant_wrapper_v**
- template<auto __cw, typename _Fn>
  constexpr bool **std::__is_constant_wrapper_v**< **constant_wrapper**< **__cw, _Fn** > >
- template<typename _Range>
  constexpr bool **std::ranges::__detail::__is_subrange**
- template<auto _Xv>
  constexpr auto **std::cw**
- constexpr _Swallow_assign std::ignore
- constexpr sorted_equivalent_t **std::sorted_equivalent**
- constexpr sorted_unique_t **std::sorted_unique**
- template<typename _Tp>
  constexpr size_t **std::tuple_size_v**

7.206.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <utility>.
This file contains the parts of `<utility>` needed by other headers, so they don't need to include the whole of `<utility>`.

7.207 valarray_after.h File Reference

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_DEFINE_EXPR_BINARY_FUNCTION**(_Fun, _UFun)
- #define **_DEFINE_EXPR_BINARY_OPERATOR**(_Op, _Name)
- #define **_DEFINE_EXPR_UNARY_FUNCTION**(_Name, _UName)
- #define **_DEFINE_EXPR_UNARY_OPERATOR**(_Op, _Name)

Functions

- template<class _Dom>
  _Expr< _UnClos< struct std::_Abs, _Expr, _Dom >, typename _Dom::value_type > **std::abs** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Abs, _ValArray, _Tp >, _Tp > **std::abs** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Acos, _Expr, _Dom >, typename _Dom::value_type > **std::acos** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Acos, _ValArray, _Tp >, _Tp > **std::acos** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Asin, _Expr, _Dom >, typename _Dom::value_type > **std::asin** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Asin, _ValArray, _Tp >, _Tp > **std::asin** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Atan, _Expr, _Dom >, typename _Dom::value_type > **std::atan** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Atan, _ValArray, _Tp >, _Tp > **std::atan** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > **std::atan2** (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > **std::atan2** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::_Atan2, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > **std::atan2** (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > **std::atan2** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Atan2, _Constant, _ValArray, _Tp, _Tp >, _Tp > **std::atan2** (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Atan2, _ValArray, _Constant, _Tp, _Tp >, _Tp > **std::atan2** (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Atan2, _ValArray, _ValArray, _Tp, _Tp >, _Tp > **std::atan2** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)

- template<class _Dom>
  _Expr< _BinClos< struct std::_Atan2, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩ Dom::value_type > **std::atan2** (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Cos, _Expr, _Dom >, typename _Dom::value_type > **std::cos** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Cos, _ValArray, _Tp >, _Tp > **std::cos** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Cosh, _Expr, _Dom >, typename _Dom::value_type > **std::cosh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Cosh, _ValArray, _Tp >, _Tp > **std::cosh** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Exp, _Expr, _Dom >, typename _Dom::value_type > **std::exp** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Exp, _ValArray, _Tp >, _Tp > **std::exp** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Log, _Expr, _Dom >, typename _Dom::value_type > **std::log** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Log, _ValArray, _Tp >, _Tp > **std::log** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Log10, _Expr, _Dom >, typename _Dom::value_type > **std::log10** (const _↩ Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Log10, _ValArray, _Tp >, _Tp > **std::log10** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **std::operator!=** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **std::operator!=** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__not_equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::↩ __not_equal_to, typename _Dom1::value_type >::result_type > **std::operator!=** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **std::operator!=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__not_equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__not_equal_to, typename _Dom::value_type >::result_type > **std::operator!=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **std::operator%** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)

- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **std::operator%** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__modulus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__modulus, typename _Dom1::value_type >::result_type > **std::operator%** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **std::operator%** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__modulus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__modulus, typename _Dom::value_type >::result_type > **std::operator%** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__bitwise_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__bitwise_and, typename _Dom1::value_type >::result_type > **std::operator&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_and, typename _Dom::value_type >::result_type > **std::operator&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **std::operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **std::operator&&** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__logical_and, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__logical_and, typename _Dom1::value_type >::result_type > **std::operator&&** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **std::operator&&** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_and, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_and, typename _Dom::value_type >::result_type > **std::operator&&** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **std::operator*** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **std::operator*** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__multiplies, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__multiplies, typename _Dom1::value_type >::result_type > **std::operator*** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **std::operator*** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__multiplies, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__multiplies, typename _Dom::value_type >::result_type > **std::operator*** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__plus, typename _Dom::value_type >::result_type > **std::operator+** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__plus, typename _Dom::value_type >::result_type > **std::operator+** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__plus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__plus, typename _Dom1::value_type >::result_type > **std::operator+** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__plus, typename _Dom::value_type >::result_type > **std::operator+** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__plus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__plus, typename _Dom::value_type >::result_type > **std::operator+** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__minus, typename _Dom::value_type >::result_type > **std::operator-** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__minus, typename _Dom::value_type >::result_type > **std::operator-** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)

- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__minus, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__↩ minus, typename _Dom1::value_type >::result_type > **std::operator-** (const _Expr< _Dom1, typename _↩ Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__minus, typename _Dom::value_type >::result_type > **std::operator-** (const typename _↩ Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__minus, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩ _fun< struct std::__minus, typename _Dom::value_type >::result_type > **std::operator-** (const valarray< type- name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _↩ _fun< struct std::__divides, typename _Dom::value_type >::result_type > **std::operator/** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename ↩ __fun< struct std::__divides, typename _Dom::value_type >::result_type > **std::operator/** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__divides, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__↩ divides, typename _Dom1::value_type >::result_type > **std::operator/** (const _Expr< _Dom1, typename _↩ Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__divides, typename _Dom::value_type >::result_type > **std::operator/** (const typename _↩ Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__divides, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩ _fun< struct std::__divides, typename _Dom::value_type >::result_type > **std::operator/** (const valarray< type- name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _↩ _fun< struct std::__less, typename _Dom::value_type >::result_type > **std::operator<** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _↩ _fun< struct std::__less, typename _Dom::value_type >::result_type > **std::operator<** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__less, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__less, typename _Dom1::value_type >::result_type > **std::operator<** (const _Expr< _Dom1, typename _Dom1::↩ value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename ↩ __fun< struct std::__less, typename _Dom::value_type >::result_type > **std::operator<** (const typename _↩ Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _↩ _fun< struct std::__less, typename _Dom::value_type >::result_type > **std::operator<** (const valarray< type- name _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **std::operator**<< (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **std::operator**<< (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__shift_left, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::_↩ _shift_left, typename _Dom1::value_type >::result_type > **std::operator**<< (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **std::operator**<< (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_left, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_left, typename _Dom::value_type >::result_type > **std::operator**<< (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **std::operator**<= (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **std::operator**<= (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__less_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__↩ less_equal, typename _Dom1::value_type >::result_type > **std::operator**<= (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, type- name __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **std::operator**<= (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__less_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, type- name __fun< struct std::__less_equal, typename _Dom::value_type >::result_type > **std::operator**<= (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **std::operator==** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **std::operator==** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__equal_to, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::_↩ _equal_to, typename _Dom1::value_type >::result_type > **std::operator==** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)

- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **std::operator==** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__equal_to, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__equal_to, typename _Dom::value_type >::result_type > **std::operator==** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > **std::operator>** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > **std::operator>** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__greater, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__greater, typename _Dom1::value_type >::result_type > **std::operator>** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > **std::operator>** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater, typename _Dom::value_type >::result_type > **std::operator>** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **std::operator>=** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **std::operator>=** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__greater_equal, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__greater_equal, typename _Dom1::value_type >::result_type > **std::operator>=** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **std::operator>=** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__greater_equal, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__greater_equal, typename _Dom::value_type >::result_type > **std::operator>=** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **std::operator>>** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)

- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **std::operator**>> (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__shift_right, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__shift_right, typename _Dom1::value_type >::result_type > **std::operator**>> (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **std::operator**>> (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__shift_right, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__shift_right, typename _Dom::value_type >::result_type > **std::operator**>> (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **std::operator^** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **std::operator^** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__bitwise_xor, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__bitwise_xor, typename _Dom1::value_type >::result_type > **std::operator^** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **std::operator^** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_xor, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_xor, typename _Dom::value_type >::result_type > **std::operator^** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **std::operator|** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **std::operator|** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__bitwise_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__bitwise_or, typename _Dom1::value_type >::result_type > **std::operator|** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **std::operator|** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)

- template<class _Dom>
  _Expr< _BinClos< struct std::__bitwise_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__bitwise_or, typename _Dom::value_type >::result_type > **std::operator|** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **std::operator||** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **std::operator||** (const _Expr< _Dom, typename _Dom::value_type > &__v, const typename _Dom::value_type &__t)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::__logical_or, _Expr, _Expr, _Dom1, _Dom2 >, typename __fun< struct std::__logical_or, typename _Dom1::value_type >::result_type > **std::operator||** (const _Expr< _Dom1, typename _Dom1::value_type > &__v, const _Expr< _Dom2, typename _Dom2::value_type > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **std::operator||** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__v)
- template<class _Dom>
  _Expr< _BinClos< struct std::__logical_or, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename __fun< struct std::__logical_or, typename _Dom::value_type >::result_type > **std::operator||** (const valarray< typename _Dom::value_type > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _Expr, _Constant, _Dom, typename _Dom::value_type >, typename _Dom::value_type > **std::pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const typename _Dom::value_type &__t)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _Expr, _ValArray, _Dom, typename _Dom::value_type >, typename _Dom::value_type > **std::pow** (const _Expr< _Dom, typename _Dom::value_type > &__e, const valarray< typename _Dom::value_type > &__v)
- template<class _Dom1, class _Dom2>
  _Expr< _BinClos< struct std::_Pow, _Expr, _Expr, _Dom1, _Dom2 >, typename _Dom1::value_type > **std::pow** (const _Expr< _Dom1, typename _Dom1::value_type > &__e1, const _Expr< _Dom2, typename _Dom2::value_type > &__e2)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _Constant, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > **std::pow** (const typename _Dom::value_type &__t, const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Pow, _Constant, _ValArray, _Tp, _Tp >, _Tp > **std::pow** (const typename valarray< _Tp >::value_type &__t, const valarray< _Tp > &__v)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Pow, _ValArray, _Constant, _Tp, _Tp >, _Tp > **std::pow** (const valarray< _Tp > &__v, const typename valarray< _Tp >::value_type &__t)
- template<typename _Tp>
  _Expr< _BinClos< struct std::_Pow, _ValArray, _ValArray, _Tp, _Tp >, _Tp > **std::pow** (const valarray< _Tp > &__v, const valarray< _Tp > &__w)
- template<class _Dom>
  _Expr< _BinClos< struct std::_Pow, _ValArray, _Expr, typename _Dom::value_type, _Dom >, typename _Dom::value_type > **std::pow** (const valarray< typename _Dom::valarray > &__v, const _Expr< _Dom, typename _Dom::value_type > &__e)

- template<class _Dom>
  _Expr< _UnClos< struct std::_Sin, _Expr, _Dom >, typename _Dom::value_type > **std::sin** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Sin, _ValArray, _Tp >, _Tp > **std::sin** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Sinh, _Expr, _Dom >, typename _Dom::value_type > **std::sinh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Sinh, _ValArray, _Tp >, _Tp > **std::sinh** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Sqrt, _Expr, _Dom >, typename _Dom::value_type > **std::sqrt** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Sqrt, _ValArray, _Tp >, _Tp > **std::sqrt** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Tan, _Expr, _Dom >, typename _Dom::value_type > **std::tan** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Tan, _ValArray, _Tp >, _Tp > **std::tan** (const valarray< _Tp > &__v)
- template<class _Dom>
  _Expr< _UnClos< struct std::_Tanh, _Expr, _Dom >, typename _Dom::value_type > **std::tanh** (const _Expr< _Dom, typename _Dom::value_type > &__e)
- template<typename _Tp>
  _Expr< _UnClos< struct std::_Tanh, _ValArray, _Tp >, _Tp > **std::tanh** (const valarray< _Tp > &__v)

7.207.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

7.208 valarray_array.h File Reference

Namespaces

- namespace std

Macros

- #define **_DEFINE_ARRAY_FUNCTION**(_Op, _Name)

Functions

- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)

- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, size_t __s1, _Array< _Tp > __b, size_t __s2)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __src, size_t __n, _Array< size_t > __i, _Array< _Tp > __dst, _Array< size_t > __j)
- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __a, _Tp *__restrict__ __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __a, const size_t *__restrict__ __i, _Tp *__restrict__ __b, size_t __n)
- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __a, size_t __n, _Tp *__restrict__ __b)
- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __a, size_t __n, _Tp *__restrict__ __b, const size_t *__restrict__ __i)
- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __a, size_t __n, size_t __s, _Tp *__restrict__ __b)
- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __src, size_t __n, const size_t *__restrict__ __i, _Tp *__restrict__ __dst, const size_t *__restrict__ __j)
- template<typename _Tp>
  void **std::__valarray_copy** (const _Tp *__restrict__ __src, size_t __n, size_t __s1, _Tp *__restrict__ __dst, size_t __s2)
- template<typename _Tp>
  void **std::__valarray_copy_construct** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **std::__valarray_copy_construct** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::__valarray_copy_construct** (const _Tp *__b, const _Tp *__e, _Tp *__restrict__ __o)
- template<typename _Tp>
  void **std::__valarray_copy_construct** (const _Tp *__restrict__ __a, const size_t *__restrict__ __i, _Tp *__restrict__ __o, size_t __n)
- template<typename _Tp>
  void **std::__valarray_copy_construct** (const _Tp *__restrict__ __a, size_t __n, size_t __s, _Tp *__restrict__ __o)
- template<typename _Tp>
  void **std::__valarray_default_construct** (_Tp *__b, _Tp *__e)
- template<typename _Tp>
  void **std::__valarray_destroy_elements** (_Tp *__b, _Tp *__e)
- template<typename _Tp>
  void **std::__valarray_fill** (_Array< _Tp > __a, _Array< size_t > __i, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::__valarray_fill** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::__valarray_fill** (_Array< _Tp > __a, size_t __n, size_t __s, const _Tp &__t)
- template<typename _Tp>
  void **std::__valarray_fill** (_Tp *__restrict__ __a, const size_t *__restrict__ __i, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::__valarray_fill** (_Tp *__restrict__ __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::__valarray_fill** (_Tp *__restrict__ __a, size_t __n, size_t __s, const _Tp &__t)

- template<typename _Tp>
  void **std::__valarray_fill_construct** (_Tp *__b, _Tp *__e, const _Tp __t)
- template<typename _Tp>
  _Tp * **std::__valarray_get_storage** (size_t)
- template<typename _Ta>
  _Ta::value_type **std::__valarray_max** (const _Ta &__a)
- template<typename _Ta>
  _Ta::value_type **std::__valarray_min** (const _Ta &__a)
- template<typename _Tp>
  void **std::__valarray_release_memory** (_Tp *__p, size_t __n)
- template<typename _Tp>
  _Tp **std::__valarray_sum** (const _Tp *__f, const _Tp *__l)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_and** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)

- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_or** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &↩ __e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)

- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___bitwise_xor** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___divides** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)

- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___minus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___modulus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)

- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___multiplies** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)

- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___plus** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_left** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< _Tp > __b, size_t __n, size_t __s)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< bool > __m, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< size_t > __i, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, _Array< size_t > __i, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, const _Expr< _Dom, _Tp > &__e, size_t __n)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b)

- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< size_t > __i)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, const _Tp &__t)
- template<typename _Tp>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __n, size_t __s, _Array< _Tp > __b)
- template<typename _Tp, class _Dom>
  void **std::_Array_augmented___shift_right** (_Array< _Tp > __a, size_t __s, const _Expr< _Dom, _Tp > &__e, size_t __n)

7.208.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

7.209 valarray_array.tcc File Reference

Namespaces

- namespace std

Functions

- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, _Array< bool > __m, size_t __n, _Array< _Tp > __b, _Array< bool > __k)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __a, size_t __n, _Array< _Tp > __b, _Array< bool > __m)
- template<typename _Tp>
  void **std::__valarray_copy** (_Array< _Tp > __e, _Array< size_t > __f, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)
- template<typename _Tp, class _Dom>
  void **std::__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)
- template<typename _Tp, class _Dom>
  void **std::__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< bool > __m)
- template<typename _Tp, class _Dom>
  void **std::__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, _Array< size_t > __i)
- template<typename _Tp, class _Dom>
  void **std::__valarray_copy** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a, size_t __s)
- template<typename _Tp>
  void **std::__valarray_copy_construct** (_Array< _Tp > __a, _Array< bool > __m, _Array< _Tp > __b, size_t __n)
- template<typename _Tp, class _Dom>
  void **std::__valarray_copy_construct** (const _Expr< _Dom, _Tp > &__e, size_t __n, _Array< _Tp > __a)
- template<typename _Tp>
  void **std::__valarray_fill** (_Array< _Tp > __a, size_t __n, _Array< bool > __m, const _Tp &__t)

7.209.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

7.210 valarray_before.h File Reference

Namespaces

- namespace std
- namespace std::__detail

7.210.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <valarray>.

7.211 vector.tcc File Reference

Namespaces

- namespace std

7.211.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <vector>.

7.212 version.h File Reference

Macros

- #define **__glibcxx_adaptor_iterator_pair_constructor**
- #define **__glibcxx_addressof_constexpr**
- #define **__glibcxx_algorithm_default_value_type**
- #define **__glibcxx_algorithm_iterator_requirements**
- #define **__glibcxx_aligned_accessor**
- #define **__glibcxx_allocate_at_least**
- #define **__glibcxx_allocator_traits_is_always_equal**
- #define **__glibcxx_any**
- #define **__glibcxx_apply**
- #define **__glibcxx_array_constexpr**
- #define **__glibcxx_as_const**
- #define **__glibcxx_associative_heterogeneous_erasure**
- #define **__glibcxx_associative_heterogeneous_insertion**
- #define **__glibcxx_assume_aligned**
- #define **__glibcxx_atomic_flag_test**
- #define **__glibcxx_atomic_float**
- #define **__glibcxx_atomic_is_always_lock_free**
- #define **__glibcxx_atomic_min_max**
- #define **__glibcxx_atomic_ref**
- #define **__glibcxx_atomic_shared_ptr**
- #define **__glibcxx_atomic_value_initialization**
- #define **__glibcxx_atomic_wait**

- #define **__glibcxx_barrier**
- #define **__glibcxx_bind_back**
- #define **__glibcxx_bind_front**
- #define **__glibcxx_bit_cast**
- #define **__glibcxx_bitops**
- #define **__glibcxx_bitset**
- #define **__glibcxx_bool_constant**
- #define **__glibcxx_bounded_array_traits**
- #define **__glibcxx_boyer_moore_searcher**
- #define **__glibcxx_byte**
- #define **__glibcxx_byteswap**
- #define **__glibcxx_chrono**
- #define **__glibcxx_chrono_cxx20**
- #define **__glibcxx_chrono_udls**
- #define **__glibcxx_clamp**
- #define **__glibcxx_common_reference**
- #define **__glibcxx_common_reference_wrapper**
- #define **__glibcxx_complex_udls**
- #define **__glibcxx_concepts**
- #define **__glibcxx_constant_wrapper**
- #define **__glibcxx_constexpr_algorithms**
- #define **__glibcxx_constexpr_bitset**
- #define **__glibcxx_constexpr_char_traits**
- #define **__glibcxx_constexpr_charconv**
- #define **__glibcxx_constexpr_complex**
- #define **__glibcxx_constexpr_dynamic_alloc**
- #define **__glibcxx_constexpr_flat_map**
- #define **__glibcxx_constexpr_flat_set**
- #define **__glibcxx_constexpr_functional**
- #define **__glibcxx_constexpr_iterator**
- #define **__glibcxx_constexpr_memory**
- #define **__glibcxx_constexpr_numeric**
- #define **__glibcxx_constexpr_string**
- #define **__glibcxx_constexpr_string_view**
- #define **__glibcxx_constexpr_tuple**
- #define **__glibcxx_constexpr_typeinfo**
- #define **__glibcxx_constexpr_utility**
- #define **__glibcxx_constexpr_vector**
- #define **__glibcxx_constrained_equality**
- #define **__glibcxx_containers_ranges**
- #define **__glibcxx_copyable_function**
- #define **__glibcxx_coroutine**
- #define **__glibcxx_debugging**
- #define **__glibcxx_enable_shared_from_this**
- #define **__glibcxx_endian**
- #define **__glibcxx_erase_if**
- #define **__glibcxx_exception_ptr_cast**
- #define **__glibcxx_exchange_function**
- #define **__glibcxx_execution**
- #define **__glibcxx_expected**
- #define **__glibcxx_filesystem**

- #define **__glibcxx_flat_map**
- #define **__glibcxx_flat_set**
- #define **__glibcxx_format**
- #define **__glibcxx_format_path**
- #define **__glibcxx_format_ranges**
- #define **__glibcxx_format_uchar**
- #define **__glibcxx_formatters**
- #define **__glibcxx_forward_like**
- #define **__glibcxx_freestanding_algorithm**
- #define **__glibcxx_freestanding_array**
- #define **__glibcxx_freestanding_cstring**
- #define **__glibcxx_freestanding_optional**
- #define **__glibcxx_freestanding_string_view**
- #define **__glibcxx_freestanding_variant**
- #define **__glibcxx_fstream_native_handle**
- #define **__glibcxx_function_ref**
- #define **__glibcxx_gcd**
- #define **__glibcxx_gcd_lcm**
- #define **__glibcxx_generator**
- #define **__glibcxx_generic_associative_lookup**
- #define **__glibcxx_generic_unordered_lookup**
- #define **__glibcxx_has_unique_object_representations**
- #define **__glibcxx_hypot**
- #define **__glibcxx_incomplete_container_elements**
- #define **__glibcxx_indirect**
- #define **__glibcxx_initializer_list**
- #define **__glibcxx_inplace_vector**
- #define **__glibcxx_int_pow2**
- #define **__glibcxx_integer_comparison_functions**
- #define **__glibcxx_integer_sequence**
- #define **__glibcxx_integral_constant_callable**
- #define **__glibcxx_interpolate**
- #define **__glibcxx_invoke**
- #define **__glibcxx_invoke_r**
- #define **__glibcxx_ios_noreplace**
- #define **__glibcxx_is_aggregate**
- #define **__glibcxx_is_constant_evaluated**
- #define **__glibcxx_is_final**
- #define **__glibcxx_is_implicit_lifetime**
- #define **__glibcxx_is_invocable**
- #define **__glibcxx_is_layout_compatible**
- #define **__glibcxx_is_nothrow_convertible**
- #define **__glibcxx_is_null_pointer**
- #define **__glibcxx_is_pointer_interconvertible**
- #define **__glibcxx_is_scoped_enum**
- #define **__glibcxx_is_structural**
- #define **__glibcxx_is_sufficiently_aligned**
- #define **__glibcxx_is_swappable**
- #define **__glibcxx_is_virtual_base_of**
- #define **__glibcxx_jthread**
- #define **__glibcxx_latch**

- #define **__glibcxx_launder**
- #define **__glibcxx_lcm**
- #define **__glibcxx_list_remove_return_type**
- #define **__glibcxx_logical_traits**
- #define **__glibcxx_make_from_tuple**
- #define **__glibcxx_make_obj_using_allocator**
- #define **__glibcxx_make_reverse_iterator**
- #define **__glibcxx_make_unique**
- #define **__glibcxx_map_try_emplace**
- #define **__glibcxx_math_constants**
- #define **__glibcxx_math_spec_funcs**
- #define **__glibcxx_math_special_functions**
- #define **__glibcxx_mdspan**
- #define **__glibcxx_memory_resource**
- #define **__glibcxx_move_iterator_concept**
- #define **__glibcxx_move_only_function**
- #define **__glibcxx_node_extract**
- #define **__glibcxx_nonmember_container_access**
- #define **__glibcxx_not_fn**
- #define **__glibcxx_null_iterators**
- #define **__glibcxx_observable_checkpoint**
- #define **__glibcxx_optional**
- #define **__glibcxx_optional_range_support**
- #define **__glibcxx_out_ptr**
- #define **__glibcxx_padded_layouts**
- #define **__glibcxx_parallel_algorithm**
- #define **__glibcxx_philox_engine**
- #define **__glibcxx_polymorphic**
- #define **__glibcxx_polymorphic_allocator**
- #define **__glibcxx_print**
- #define **__glibcxx_quoted_string_io**
- #define **__glibcxx_ranges**
- #define **__glibcxx_ranges_as_const**
- #define **__glibcxx_ranges_as_input**
- #define **__glibcxx_ranges_as_rvalue**
- #define **__glibcxx_ranges_cache_latest**
- #define **__glibcxx_ranges_cartesian_product**
- #define **__glibcxx_ranges_chunk**
- #define **__glibcxx_ranges_chunk_by**
- #define **__glibcxx_ranges_contains**
- #define **__glibcxx_ranges_enumerate**
- #define **__glibcxx_ranges_filter**
- #define **__glibcxx_ranges_find_last**
- #define **__glibcxx_ranges_fold**
- #define **__glibcxx_ranges_indices**
- #define **__glibcxx_ranges_iota**
- #define **__glibcxx_ranges_join_with**
- #define **__glibcxx_ranges_repeat**
- #define **__glibcxx_ranges_slide**
- #define **__glibcxx_ranges_starts_ends_with**
- #define **__glibcxx_ranges_stride**

- #define **__glibcxx_ranges_to_container**
- #define **__glibcxx_ranges_zip**
- #define **__glibcxx_ratio**
- #define **__glibcxx_raw_memory_algorithms**
- #define **__glibcxx_reference_from_temporary**
- #define **__glibcxx_reference_wrapper**
- #define **__glibcxx_remove_cvref**
- #define **__glibcxx_result_of_sfinae**
- #define **__glibcxx_robust_nonmodifying_seq_ops**
- #define **__glibcxx_sample**
- #define **__glibcxx_saturation_arithmetic**
- #define **__glibcxx_scoped_lock**
- #define **__glibcxx_semaphore**
- #define **__glibcxx_shared_mutex**
- #define **__glibcxx_shared_ptr_arrays**
- #define **__glibcxx_shared_ptr_weak_type**
- #define **__glibcxx_shared_timed_mutex**
- #define **__glibcxx_shift**
- #define **__glibcxx_smart_ptr_for_overwrite**
- #define **__glibcxx_smart_ptr_owner_equality**
- #define **__glibcxx_source_location**
- #define **__glibcxx_span**
- #define **__glibcxx_spanstream**
- #define **__glibcxx_ssize**
- #define **__glibcxx_sstream_from_string_view**
- #define **__glibcxx_start_lifetime_as**
- #define **__glibcxx_starts_ends_with**
- #define **__glibcxx_stdatomic_h**
- #define **__glibcxx_stdbit_h**
- #define **__glibcxx_stdckdint_h**
- #define **__glibcxx_string_contains**
- #define **__glibcxx_string_resize_and_overwrite**
- #define **__glibcxx_string_subview**
- #define **__glibcxx_string_udls**
- #define **__glibcxx_string_view**
- #define **__glibcxx_submdspan**
- #define **__glibcxx_three_way_comparison**
- #define **__glibcxx_to_address**
- #define **__glibcxx_to_underlying**
- #define **__glibcxx_transformation_trait_aliases**
- #define **__glibcxx_transparent_operators**
- #define **__glibcxx_tuple_element_t**
- #define **__glibcxx_tuple_like**
- #define **__glibcxx_tuples_by_type**
- #define **__glibcxx_type_identity**
- #define **__glibcxx_type_trait_variable_templates**
- #define **__glibcxx_uncaught_exceptions**
- #define **__glibcxx_unordered_map_try_emplace**
- #define **__glibcxx_unreachable**
- #define **__glibcxx_unwrap_ref**
- #define **__glibcxx_valarray**
- #define **__glibcxx_variant**
- #define **__glibcxx_void_t**

7.212.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <version>.

7.213 bitset File Reference

Classes

- struct std::_Base_bitset< _Nw >
- struct std::_Base_bitset< 0 >
- struct std::_Base_bitset< 1 >
- class std::bitset< _Nb >
- struct std::hash<::bitset< _Nb > >
- class std::bitset< _Nb >::reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_BITSET_BITS_PER_ULL**
- #define **_GLIBCXX_BITSET_BITS_PER_WORD**
- #define **_GLIBCXX_BITSET_WORDS**(__n)

Typedefs

- template<typename _CharT>
  using **std::__bitset::__string**

Functions

- template<size_t _Nb>
  constexpr bitset< _Nb > std::operator& (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  constexpr bitset< _Nb > std::operator^ (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  constexpr bitset< _Nb > std::operator| (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept

- template<class _CharT, class _Traits, size_t _Nb>
  std::basic_ostream< _CharT, _Traits > & std::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)
- template<class _CharT, class _Traits, size_t _Nb>
  std::basic_istream< _CharT, _Traits > & std::operator>> (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)

7.213.1 Detailed Description

This is a Standard C++ Library header.

7.214 debug/bitset File Reference

Classes

- class std::__debug::bitset< _Nb >
- struct std::hash< __debug::bitset< _Nb > >

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<size_t _Nb>
  constexpr bitset< _Nb > **std::__debug::operator&** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<typename _CharT, typename _Traits, size_t _Nb>
  std::basic_ostream< _CharT, _Traits > & **std::__debug::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const bitset< _Nb > &__x)
- template<typename _CharT, typename _Traits, size_t _Nb>
  std::basic_istream< _CharT, _Traits > & **std::__debug::operator>>** (std::basic_istream< _CharT, _Traits > &__is, bitset< _Nb > &__x)
- template<size_t _Nb>
  constexpr bitset< _Nb > **std::__debug::operator^** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept
- template<size_t _Nb>
  constexpr bitset< _Nb > **std::__debug::operator|** (const bitset< _Nb > &__x, const bitset< _Nb > &__y) noexcept

7.214.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.215 cassert File Reference

7.215.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `assert.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.216 ccomplex File Reference

Macros

- #define **_GLIBCXX_CCOMPLEX**

7.216.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

7.217 tr1/ccomplex File Reference

Macros

- #define **_GLIBCXX_TR1_CCOMPLEX**

7.217.1 Detailed Description

This is a TR1 C++ Library header.

7.218 cctype File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CCTYPE**

7.218.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `ctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.219 tr1/cctype File Reference

Macros

- #define **_GLIBCXX_TR1_CCTYPE**

7.219.1 Detailed Description

This is a TR1 C++ Library header.

7.220 cerrno File Reference

Macros

- #define **_GLIBCXX_CERRNO**
- #define **errno**

7.220.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `errno.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.221 cfenv File Reference

Macros

- #define **_GLIBCXX_CFENV**

7.221.1 Detailed Description

This is a Standard C++ Library header.

7.222 tr1/cfenv File Reference

Macros

- #define **_GLIBCXX_TR1_CFENV**

7.222.1 Detailed Description

This is a TR1 C++ Library header.

7.223 cfloat File Reference

Macros

- #define **_GLIBCXX_CFLOAT**
- #define **DECIMAL_DIG**
- #define **FLT_EVAL_METHOD**

7.223.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `float.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.224 tr1/cfloat File Reference

Macros

- #define **_GLIBCXX_TR1_CFLOAT**

7.224.1 Detailed Description

This is a TR1 C++ Library header.

7.225 charconv File Reference

Classes

- struct std::from_chars_result
- struct std::to_chars_result

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_GLIBCXX_TO_CHARS**(T)

Typedefs

- template<typename _Tp>
  using **std::__detail::__unsigned_least_t**

Enumerations

- enum class std::chars_format { **scientific** , **fixed** , **hex** , **general** }

Functions

- template<bool _DecOnly, typename _Tp>
  constexpr bool std::__detail::__from_chars_alnum (const char *&__first, const char *__last, _Tp &__val, int __base)
- template<bool _DecOnly = false>
  constexpr unsigned char **std::__detail::__from_chars_alnum_to_val** (unsigned char __c)
- from_chars_result **std::__from_chars_bfloat16_t** (const char *__first, const char *__last, float &__value, chars_format __fmt=chars_format::general) noexcept
- from_chars_result **std::__from_chars_float16_t** (const char *__first, const char *__last, float &__value, chars_format __fmt=chars_format::general) noexcept
- template<bool _DecOnly, typename _Tp>
  constexpr bool std::__detail::__from_chars_pow2_base (const char *&__first, const char *__last, _Tp &__val, int __base)
- template<typename _Tp>
  constexpr bool **std::__detail::__raise_and_add** (_Tp &__val, int __base, unsigned char __c)
- template<typename _Tp>
  constexpr to_chars_result **std::__detail::__to_chars** (char *__first, char *__last, _Tp __val, int __base) noexcept
- template<typename _Tp>
  constexpr to_chars_result **std::__detail::__to_chars_10** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **std::__detail::__to_chars_16** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **std::__detail::__to_chars_2** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **std::__detail::__to_chars_8** (char *__first, char *__last, _Tp __val) noexcept
- template<typename _Tp>
  constexpr to_chars_result **std::__to_chars_i** (char *__first, char *__last, _Tp __value, int __base=10)
- template<typename _Tp>
  constexpr unsigned **std::__detail::__to_chars_len** (_Tp __value, int __base) noexcept
- template<typename _Tp>
  constexpr unsigned **std::__detail::__to_chars_len_2** (_Tp __value) noexcept
- template<typename _Tp, enable_if_t< __or_< __is_signed_or_unsigned_integer< _Tp >, is_same< char, _Tp > >::value, int > = 0>
  constexpr from_chars_result std::from_chars (const char *__first, const char *__last, _Tp &__value, int __base=10)
- from_chars_result **std::from_chars** (const char *__first, const char *__last, double &__value, chars_format __fmt=chars_format::general) noexcept
- from_chars_result **std::from_chars** (const char *__first, const char *__last, float &__value, chars_format __fmt=chars_format::general) noexcept
- from_chars_result **std::from_chars** (const char *__first, const char *__last, long double &__value, chars_format __fmt=chars_format::general) noexcept
- constexpr chars_format **std::operator&** (chars_format __lhs, chars_format __rhs) noexcept
- constexpr chars_format & **std::operator&=** (chars_format &__lhs, chars_format __rhs) noexcept
- constexpr chars_format **std::operator^** (chars_format __lhs, chars_format __rhs) noexcept

- constexpr chars_format & **std::operator^=** (chars_format &__lhs, chars_format __rhs) noexcept
- constexpr chars_format **std::operator|** (chars_format __lhs, chars_format __rhs) noexcept
- constexpr chars_format & **std::operator|=** (chars_format &__lhs, chars_format __rhs) noexcept
- constexpr chars_format **std::operator~** (chars_format __fmt) noexcept
- to_chars_result **std::to_chars** (char *, char *, bool, int=10)=delete
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, char __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, signed char __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, signed int __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, signed long __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, signed long long __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, signed short __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, unsigned char __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, unsigned int __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, unsigned long __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, unsigned long long __value, int __base=10)
- constexpr to_chars_result **std::to_chars** (char *__first, char *__last, unsigned short __value, int __base=10)

7.225.1 Detailed Description

This is a Standard C++ Library header.

7.226 chrono File Reference

Classes

- class std::chrono::tzdb_list::const_iterator
- class std::chrono::gps_clock
- class std::chrono::hh_mm_ss< _Duration >
- class std::chrono::tai_clock
- class std::chrono::tzdb_list
- class std::chrono::utc_clock

Namespaces

- namespace std
- namespace std::chrono
- namespace std::literals
- namespace std::literals::chrono_literals

Typedefs

- using **std::chrono::gps_seconds**
- template<typename _Duration>
  using **std::chrono::gps_time**
- using **std::chrono::local_days**
- using **std::chrono::local_seconds**
- template<typename _Duration>
  using **std::chrono::local_time**
- using **std::chrono::tai_seconds**
- template<typename _Duration>
  using **std::chrono::tai_time**
- using **std::chrono::utc_seconds**
- template<typename _Duration>
  using **std::chrono::utc_time**
- using **std::chrono::zoned_seconds**

Enumerations

- enum class **choose** { **earliest** , **latest** }

Functions

- leap_second_info **std::chrono::__detail::__get_leap_second_info** (sys_seconds __ss, bool __is_utc)
- template<typename _DestClock, typename _SourceClock, typename _Duration>
  requires __detail::__clock_convs<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys<_DestClock, _SourceClock, ↩ _Duration> || __detail::__clock_convs_utc<_DestClock, _SourceClock, _Duration> || __detail::__clock_convs_sys_utc<_DestClock, _↩ SourceClock, _Duration> || __detail::__clock_convs_utc_sys<_DestClock, _SourceClock, _Duration>
  auto std::chrono::clock_cast (const time_point< _SourceClock, _Duration > &__t)
- const time_zone * **std::chrono::current_zone** ()
- template<typename _Duration>
  leap_second_info **std::chrono::get_leap_second_info** (const utc_time< _Duration > &__ut)
- const tzdb & **std::chrono::get_tzdb** ()
- tzdb_list & **std::chrono::get_tzdb_list** ()
- constexpr bool std::chrono::is_am (const hours &__h) noexcept
- constexpr bool std::chrono::is_pm (const hours &__h) noexcept
- const time_zone * **std::chrono::locate_zone** (string_view __tz_name)
- constexpr hours std::chrono::make12 (const hours &__h) noexcept
- constexpr hours std::chrono::make24 (const hours &__h, bool __is_pm) noexcept
- constexpr chrono::day std::literals::chrono_literals::operator""d (unsigned long long __d) noexcept
- constexpr chrono::year std::literals::chrono_literals::operator""y (unsigned long long __y) noexcept
- template<typename _Dur1, typename _TZPtr1, typename _Dur2, typename _TZPtr2>
  bool **std::chrono::operator==** (const zoned_time< _Dur1, _TZPtr1 > &__x, const zoned_time< _Dur2, _TZPtr2 > &__y)
- const tzdb & **std::chrono::reload_tzdb** ()
- string **std::chrono::remote_version** ()
- **std::chrono::zoned_time** () -> zoned_time< seconds >
- template<typename _TimeZonePtrOrName>
  std::chrono::zoned_time (_TimeZonePtrOrName &&) -> zoned_time< seconds, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _TimeZonePtrOrName, typename _Duration>
  std::chrono::zoned_time (_TimeZonePtrOrName &&, local_time< _Duration >, choose=choose::earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _TimeZonePtrOrName, typename _Duration>
  std::chrono::zoned_time (_TimeZonePtrOrName &&, sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZonePtrOrName > >
- template<typename _Duration, typename _TimeZonePtrOrName, typename _TimeZonePtr2>
  std::chrono::zoned_time (_TimeZonePtrOrName &&, zoned_time< _Duration, _TimeZonePtr2 >, choose=choose::↩ earliest) -> zoned_time< common_type_t< _Duration, seconds >, __time_zone_representation< _TimeZone↩ PtrOrName > >
- template<typename _Duration>
  std::chrono::zoned_time (sys_time< _Duration >) -> zoned_time< common_type_t< _Duration, seconds > >

Variables

- constexpr month **std::chrono::April**
- constexpr month **std::chrono::August**
- constexpr month **std::chrono::December**

- constexpr month **std::chrono::February**
- constexpr weekday **std::chrono::Friday**
- template<> constexpr bool **std::chrono::is_clock_v**< **gps_clock** >
- template<> constexpr bool **std::chrono::is_clock_v**< **tai_clock** >
- template<> constexpr bool **std::chrono::is_clock_v**< **utc_clock** >
- constexpr month **std::chrono::January**
- constexpr month **std::chrono::July**
- constexpr month **std::chrono::June**
- constexpr last_spec **std::chrono::last**
- constexpr month **std::chrono::March**
- constexpr month **std::chrono::May**
- constexpr weekday **std::chrono::Monday**
- constexpr month **std::chrono::November**
- constexpr month **std::chrono::October**
- constexpr weekday **std::chrono::Saturday**
- constexpr month **std::chrono::September**
- constexpr weekday **std::chrono::Sunday**
- constexpr weekday **std::chrono::Thursday**
- constexpr weekday **std::chrono::Tuesday**
- constexpr weekday **std::chrono::Wednesday**

7.226.1 Detailed Description

This is a Standard C++ Library header.

7.227 experimental/chrono File Reference

Namespaces

- namespace std
- namespace std::chrono

Macros

- #define **_GLIBCXX_EXPERIMENTAL_CHRONO**

Variables

- template<typename _Rep>
  constexpr bool **std::chrono::experimental::treat_as_floating_point_v**

7.227.1 Detailed Description

This is a TS C++ Library header.

7.228 cinttypes File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CINTTYPES**

7.228.1 Detailed Description

This is a Standard C++ Library header.

7.229 tr1/cinttypes File Reference

Macros

- #define **_GLIBCXX_TR1_CINTTYPES**

7.229.1 Detailed Description

This is a TR1 C++ Library header.

7.230 ciso646 File Reference

7.230.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `iso646.h`, which is empty in C++.

Since

C++11 (removed in C++20)

7.231 climits File Reference

Macros

- #define **_GLIBCXX_CLIMITS**
- #define **LLONG_MAX**
- #define **LLONG_MIN**
- #define **ULLONG_MAX**

7.231.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `limits.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.232 tr1/climits File Reference

Macros

- #define **_GLIBCXX_TR1_CLIMITS**

7.232.1 Detailed Description

This is a TR1 C++ Library header.

7.233 clocale File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CLOCALE**

7.233.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `locale.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.234 cmath File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CMATH**
- #define **_GLIBCXX_INCLUDE_NEXT_C_HEADERS**

Functions

- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::acos** (_Tp __x)
- constexpr float **std::acos** (float __x)
- constexpr long double **std::acos** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::acosh** (_Tp __x)
- constexpr float std::acosh (float __x)
- constexpr long double **std::acosh** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::asin** (_Tp __x)
- constexpr float **std::asin** (float __x)
- constexpr long double **std::asin** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::asinh** (_Tp __x)
- constexpr float **std::asinh** (float __x)
- constexpr long double **std::asinh** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::atan** (_Tp __x)
- constexpr float **std::atan** (float __x)
- constexpr long double **std::atan** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::atan2** (_Tp __y, _Up __x)
- constexpr float **std::atan2** (float __y, float __x)
- constexpr long double **std::atan2** (long double __y, long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::atanh** (_Tp __x)
- constexpr float **std::atanh** (float __x)
- constexpr long double **std::atanh** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::cbrt** (_Tp __x)

- constexpr float **std::cbrt** (float __x)
- constexpr long double **std::cbrt** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::ceil** (_Tp __x)
- constexpr float **std::ceil** (float __x)
- constexpr long double **std::ceil** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::copysign** (_Tp __x, _Up __y)
- constexpr float **std::copysign** (float __x, float __y)
- constexpr long double **std::copysign** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::cos** (_Tp __x)
- constexpr float **std::cos** (float __x)
- constexpr long double **std::cos** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::cosh** (_Tp __x)
- constexpr float **std::cosh** (float __x)
- constexpr long double **std::cosh** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::erf** (_Tp __x)
- constexpr float **std::erf** (float __x)
- constexpr long double **std::erf** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::erfc** (_Tp __x)
- constexpr float **std::erfc** (float __x)
- constexpr long double **std::erfc** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::exp** (_Tp __x)
- constexpr float **std::exp** (float __x)
- constexpr long double **std::exp** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::exp2** (_Tp __x)
- constexpr float **std::exp2** (float __x)
- constexpr long double **std::exp2** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::expm1** (_Tp __x)
- constexpr float **std::expm1** (float __x)
- constexpr long double **std::expm1** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::fabs** (_Tp __x)
- constexpr float **std::fabs** (float __x)
- constexpr long double **std::fabs** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::fdim** (_Tp __x, _Up __y)
- constexpr float **std::fdim** (float __x, float __y)
- constexpr long double **std::fdim** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::floor** (_Tp __x)
- constexpr float **std::floor** (float __x)
- constexpr long double **std::floor** (long double __x)
- template<typename _Tp, typename _Up, typename _Vp>
  constexpr __gnu_cxx::__promote_3< _Tp, _Up, _Vp >::__type **std::fma** (_Tp __x, _Up __y, _Vp __z)

- constexpr float **std::fma** (float __x, float __y, float __z)
- constexpr long double **std::fma** (long double __x, long double __y, long double __z)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::fmax** (_Tp __x, _Up __y)
- constexpr float **std::fmax** (float __x, float __y)
- constexpr long double **std::fmax** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::fmin** (_Tp __x, _Up __y)
- constexpr float **std::fmin** (float __x, float __y)
- constexpr long double **std::fmin** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::fmod** (_Tp __x, _Up __y)
- constexpr float **std::fmod** (float __x, float __y)
- constexpr long double **std::fmod** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, int >::__type **std::fpclassify** (_Tp __x)
- constexpr int **std::fpclassify** (double __x)
- constexpr int **std::fpclassify** (float __x)
- constexpr int **std::fpclassify** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::frexp** (_Tp __x, int *__exp)
- float **std::frexp** (float __x, int *__exp)
- long double **std::frexp** (long double __x, int *__exp)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::hypot** (_Tp __x, _Up __y)
- constexpr float **std::hypot** (float __x, float __y)
- constexpr long double **std::hypot** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, int >::__type **std::ilogb** (_Tp __x)
- constexpr int **std::ilogb** (float __x)
- constexpr int **std::ilogb** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **std::isfinite** (_Tp)
- constexpr bool **std::isfinite** (double __x)
- constexpr bool **std::isfinite** (float __x)
- constexpr bool **std::isfinite** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **std::isgreater** (_Tp __x, _Up __y)
- constexpr bool **std::isgreater** (double __x, double __y)
- constexpr bool **std::isgreater** (float __x, float __y)
- constexpr bool **std::isgreater** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **std::isgreaterequal** (_Tp __x, _Up __y)
- constexpr bool **std::isgreaterequal** (double __x, double __y)
- constexpr bool **std::isgreaterequal** (float __x, float __y)
- constexpr bool **std::isgreaterequal** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **std::isinf** (_Tp)
- constexpr bool **std::isinf** (double __x)

- constexpr bool **std::isinf** (float __x)
- constexpr bool **std::isinf** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **std::isless** (_Tp __x, _Up __y)
- constexpr bool **std::isless** (double __x, double __y)
- constexpr bool **std::isless** (float __x, float __y)
- constexpr bool **std::isless** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **std::islessequal** (_Tp __x, _Up __y)
- constexpr bool **std::islessequal** (double __x, double __y)
- constexpr bool **std::islessequal** (float __x, float __y)
- constexpr bool **std::islessequal** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **std::islessgreater** (_Tp __x, _Up __y)
- constexpr bool **std::islessgreater** (double __x, double __y)
- constexpr bool **std::islessgreater** (float __x, float __y)
- constexpr bool **std::islessgreater** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **std::isnan** (_Tp)
- constexpr bool **std::isnan** (double __x)
- constexpr bool **std::isnan** (float __x)
- constexpr bool **std::isnan** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **std::isnormal** (_Tp __x)
- constexpr bool **std::isnormal** (double __x)
- constexpr bool **std::isnormal** (float __x)
- constexpr bool **std::isnormal** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__enable_if<(__is_arithmetic< _Tp >::__value &&__is_arithmetic< _Up >::__value), bool >::__type **std::isunordered** (_Tp __x, _Up __y)
- constexpr bool **std::isunordered** (double __x, double __y)
- constexpr bool **std::isunordered** (float __x, float __y)
- constexpr bool **std::isunordered** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::ldexp** (_Tp __x, int __exp)
- constexpr float **std::ldexp** (float __x, int __exp)
- constexpr long double **std::ldexp** (long double __x, int __exp)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::lgamma** (_Tp __x)
- constexpr float **std::lgamma** (float __x)
- constexpr long double **std::lgamma** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, longlong >::__type **std::llrint** (_Tp __x)
- constexpr long long **std::llrint** (float __x)
- constexpr long long **std::llrint** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, longlong >::__type **std::llround** (_Tp __x)
- constexpr long long **std::llround** (float __x)

- constexpr long long **std::llround** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::log** (_Tp __x)
- constexpr float **std::log** (float __x)
- constexpr long double **std::log** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::log10** (_Tp __x)
- constexpr float **std::log10** (float __x)
- constexpr long double **std::log10** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::log1p** (_Tp __x)
- constexpr float **std::log1p** (float __x)
- constexpr long double **std::log1p** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::log2** (_Tp __x)
- constexpr float **std::log2** (float __x)
- constexpr long double **std::log2** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::logb** (_Tp __x)
- constexpr float **std::logb** (float __x)
- constexpr long double **std::logb** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, long >::__type **std::lrint** (_Tp __x)
- constexpr long **std::lrint** (float __x)
- constexpr long **std::lrint** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, long >::__type **std::lround** (_Tp __x)
- constexpr long **std::lround** (float __x)
- constexpr long **std::lround** (long double __x)
- float **std::modf** (float __x, float *__iptr)
- long double **std::modf** (long double __x, long double *__iptr)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::nearbyint** (_Tp __x)
- constexpr float **std::nearbyint** (float __x)
- constexpr long double **std::nearbyint** (long double __x)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::nextafter** (_Tp __x, _Up __y)
- constexpr float **std::nextafter** (float __x, float __y)
- constexpr long double **std::nextafter** (long double __x, long double __y)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::nexttoward** (_Tp ↩
  __x, long double __y)
- constexpr float **std::nexttoward** (float __x, long double __y)
- constexpr long double **std::nexttoward** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::pow** (_Tp __x, _Up __y)
- constexpr float **std::pow** (float __x, float __y)
- constexpr long double **std::pow** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  constexpr __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::remainder** (_Tp __x, _Up __y)
- constexpr float **std::remainder** (float __x, float __y)

- constexpr long double **std::remainder** (long double __x, long double __y)
- template<typename _Tp, typename _Up>
  __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::remquo** (_Tp __x, _Up __y, int *__pquo)
- float **std::remquo** (float __x, float __y, int *__pquo)
- long double **std::remquo** (long double __x, long double __y, int *__pquo)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::rint** (_Tp __x)
- constexpr float **std::rint** (float __x)
- constexpr long double **std::rint** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::round** (_Tp __x)
- constexpr float **std::round** (float __x)
- constexpr long double **std::round** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::scalbln** (_Tp __x, long __ex)
- constexpr float **std::scalbln** (float __x, long __ex)
- constexpr long double **std::scalbln** (long double __x, long __ex)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::scalbn** (_Tp __x, int __ex)
- constexpr float **std::scalbn** (float __x, int __ex)
- constexpr long double **std::scalbn** (long double __x, int __ex)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, bool >::__type **std::signbit** (_Tp __x)
- constexpr bool **std::signbit** (double __x)
- constexpr bool **std::signbit** (float __x)
- constexpr bool **std::signbit** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::sin** (_Tp __x)
- constexpr float **std::sin** (float __x)
- constexpr long double **std::sin** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::sinh** (_Tp __x)
- constexpr float **std::sinh** (float __x)
- constexpr long double **std::sinh** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::sqrt** (_Tp __x)
- constexpr float **std::sqrt** (float __x)
- constexpr long double **std::sqrt** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::tan** (_Tp __x)
- constexpr float **std::tan** (float __x)
- constexpr long double **std::tan** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::tanh** (_Tp __x)
- constexpr float **std::tanh** (float __x)
- constexpr long double **std::tanh** (long double __x)
- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::tgamma** (_Tp __x)
- constexpr float **std::tgamma** (float __x)
- constexpr long double **std::tgamma** (long double __x)

- template<typename _Tp>
  constexpr __gnu_cxx::__enable_if< __is_integer< _Tp >::__value, double >::__type **std::trunc** (_Tp __x)
- constexpr float **std::trunc** (float __x)
- constexpr long double **std::trunc** (long double __x)

7.234.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `math.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.235 ext/cmath File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_CMATH**

Variables

- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__e**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__gamma_e**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__ln_10**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__ln_2**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__ln_3**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__log10_e**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__log2_e**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__one_div_e**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__one_div_pi**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__one_div_root_2**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__phi**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__pi**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__pi_half**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__pi_quarter**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__pi_third**

- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__root_2**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__root_3**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__root_5**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__root_7**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__root_pi_div_2**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__two_div_pi**
- template<typename _RealType>
  constexpr _RealType **__gnu_cxx::__math_constants< _RealType >::__two_div_root_pi**

7.235.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.236 tr1/cmath File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_CMATH**

Functions

- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_laguerre (unsigned int __n, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_laguerref** (unsigned int __n, unsigned int __m, float __x)
- long double **std::tr1::assoc_laguerrel** (unsigned int __n, unsigned int __m, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::assoc_legendre (unsigned int __l, unsigned int __m, _Tp __x)
- float **std::tr1::assoc_legendref** (unsigned int __l, unsigned int __m, float __x)
- long double **std::tr1::assoc_legendrel** (unsigned int __l, unsigned int __m, long double __x)
- template<typename _Tpx, typename _Tpy>
  __gnu_cxx::__promote_2< _Tpx, _Tpy >::__type std::tr1::beta (_Tpx __x, _Tpy __y)
- float **std::tr1::betaf** (float __x, float __y)
- long double **std::tr1::betal** (long double __x, long double __y)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_1 (_Tp __k)
- float **std::tr1::comp_ellint_1f** (float __k)
- long double **std::tr1::comp_ellint_1l** (long double __k)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::comp_ellint_2 (_Tp __k)
- float **std::tr1::comp_ellint_2f** (float __k)
- long double **std::tr1::comp_ellint_2l** (long double __k)
- template<typename _Tp, typename _Tpn>
  __gnu_cxx::__promote_2< _Tp, _Tpn >::__type std::tr1::comp_ellint_3 (_Tp __k, _Tpn __nu)

- float **std::tr1::comp_ellint_3f** (float __k, float __nu)
- long double **std::tr1::comp_ellint_3l** (long double __k, long double __nu)
- template<typename _Tpa, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_3< _Tpa, _Tpc, _Tp >::__type std::tr1::conf_hyperg (_Tpa __a, _Tpc __c, _Tp __x)
- float std::tr1::conf_hypergf (float __a, float __c, float __x)
- long double std::tr1::conf_hypergl (long double __a, long double __c, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_i (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_if** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_il** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_j (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_jf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_jl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_bessel_k (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_bessel_kf** (float __nu, float __x)
- long double **std::tr1::cyl_bessel_kl** (long double __nu, long double __x)
- template<typename _Tpnu, typename _Tp>
  __gnu_cxx::__promote_2< _Tpnu, _Tp >::__type std::tr1::cyl_neumann (_Tpnu __nu, _Tp __x)
- float **std::tr1::cyl_neumannf** (float __nu, float __x)
- long double **std::tr1::cyl_neumannl** (long double __nu, long double __x)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_1 (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_1f** (float __k, float __phi)
- long double **std::tr1::ellint_1l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpp>
  __gnu_cxx::__promote_2< _Tp, _Tpp >::__type std::tr1::ellint_2 (_Tp __k, _Tpp __phi)
- float **std::tr1::ellint_2f** (float __k, float __phi)
- long double **std::tr1::ellint_2l** (long double __k, long double __phi)
- template<typename _Tp, typename _Tpn, typename _Tpp>
  __gnu_cxx::__promote_3< _Tp, _Tpn, _Tpp >::__type std::tr1::ellint_3 (_Tp __k, _Tpn __nu, _Tpp __phi)
- float **std::tr1::ellint_3f** (float __k, float __nu, float __phi)
- long double **std::tr1::ellint_3l** (long double __k, long double __nu, long double __phi)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::expint (_Tp __x)
- float **std::tr1::expintf** (float __x)
- long double **std::tr1::expintl** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type **std::tr1::fabs** (_Tp __x)
- float **std::tr1::fabs** (float __x)
- long double **std::tr1::fabs** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::hermite (unsigned int __n, _Tp __x)
- float **std::tr1::hermitef** (unsigned int __n, float __x)
- long double **std::tr1::hermitel** (unsigned int __n, long double __x)
- template<typename _Tpa, typename _Tpb, typename _Tpc, typename _Tp>
  __gnu_cxx::__promote_4< _Tpa, _Tpb, _Tpc, _Tp >::__type std::tr1::hyperg (_Tpa __a, _Tpb __b, _Tpc __c, _Tp __x)
- float std::tr1::hypergf (float __a, float __b, float __c, float __x)
- long double std::tr1::hypergl (long double __a, long double __b, long double __c, long double __x)

- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::laguerre (unsigned int __n, _Tp __x)
- float **std::tr1::laguerref** (unsigned int __n, float __x)
- long double **std::tr1::laguerrel** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::legendre (unsigned int __n, _Tp __x)
- float **std::tr1::legendref** (unsigned int __n, float __x)
- long double **std::tr1::legendrel** (unsigned int __n, long double __x)
- template<typename _Tp, typename _Up>
  __gnu_cxx::__promote_2< _Tp, _Up >::__type **std::tr1::pow** (_Tp __x, _Up __y)
- float **std::tr1::pow** (float __x, float __y)
- long double **std::tr1::pow** (long double __x, long double __y)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::riemann_zeta (_Tp __x)
- float **std::tr1::riemann_zetaf** (float __x)
- long double **std::tr1::riemann_zetal** (long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_bessel (unsigned int __n, _Tp __x)
- float **std::tr1::sph_besself** (unsigned int __n, float __x)
- long double **std::tr1::sph_bessell** (unsigned int __n, long double __x)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_legendre (unsigned int __l, unsigned int __m, _Tp __theta)
- float **std::tr1::sph_legendref** (unsigned int __l, unsigned int __m, float __theta)
- long double **std::tr1::sph_legendrel** (unsigned int __l, unsigned int __m, long double __theta)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type std::tr1::sph_neumann (unsigned int __n, _Tp __x)
- float **std::tr1::sph_neumannf** (unsigned int __n, float __x)
- long double **std::tr1::sph_neumannl** (unsigned int __n, long double __x)

7.236.1 Detailed Description

This is a TR1 C++ Library header.

7.237 codecvt File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CODECVT_SPECIALIZATION**(_NAME, _ELEM)
- #define **_GLIBCXX_CODECVT_SPECIALIZATION2**(_NAME, _ELEM)

Enumerations

- enum **codecvt_mode** { **consume_header** , **generate_header** , **little_endian** }

7.237.1 Detailed Description

This is a Standard C++ Library header.

7.238 complex File Reference

Classes

- class std::complex< _Tp >
- class std::complex< double >
- class std::complex< float >
- class std::complex< long double >

Namespaces

- namespace std

Functions

- template<typename _Tp>
  _Tp **std::__complex_abs** (const complex< _Tp > &__z)
- __complex__ double **std::__complex_acos** (__complex__ double __z)
- __complex__ float **std::__complex_acos** (__complex__ float __z)
- __complex__ long double **std::__complex_acos** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_acos** (const std::complex< _Tp > &__z)
- __complex__ double **std::__complex_acosh** (__complex__ double __z)
- __complex__ float **std::__complex_acosh** (__complex__ float __z)
- __complex__ long double **std::__complex_acosh** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_acosh** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  _Tp **std::__complex_arg** (const complex< _Tp > &__z)
- __complex__ double **std::__complex_asin** (__complex__ double __z)
- __complex__ float **std::__complex_asin** (__complex__ float __z)
- __complex__ long double **std::__complex_asin** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_asin** (const std::complex< _Tp > &__z)
- __complex__ double **std::__complex_asinh** (__complex__ double __z)
- __complex__ float **std::__complex_asinh** (__complex__ float __z)
- __complex__ long double **std::__complex_asinh** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_asinh** (const std::complex< _Tp > &__z)
- __complex__ double **std::__complex_atan** (__complex__ double __z)
- __complex__ float **std::__complex_atan** (__complex__ float __z)
- __complex__ long double **std::__complex_atan** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_atan** (const std::complex< _Tp > &__z)
- __complex__ double **std::__complex_atanh** (__complex__ double __z)
- __complex__ float **std::__complex_atanh** (__complex__ float __z)
- __complex__ long double **std::__complex_atanh** (const __complex__ long double &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_atanh** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_cos** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_cosh** (const complex< _Tp > &__z)

- template<typename _Tp>
  complex< _Tp > **std::__complex_exp** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_log** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_pow** (const complex< _Tp > &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  complex< _Tp > **std::__complex_pow_unsigned** (complex< _Tp > __x, unsigned __n)
- complex< double > **std::__complex_proj** (const complex< double > &__z)
- complex< float > **std::__complex_proj** (const complex< float > &__z)
- complex< long double > **std::__complex_proj** (const complex< long double > &__z)
- template<typename _Tp>
  std::complex< _Tp > **std::__complex_proj** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_sin** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_sinh** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_sqrt** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_tan** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > **std::__complex_tanh** (const complex< _Tp > &__z)
- template<typename _Tp>
  _Tp std::abs (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::acos** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::acosh** (const std::complex< _Tp > &)
- template<typename _Tp>
  __gnu_cxx::__promote< _Tp >::__type **std::arg** (_Tp __x)
- template<typename _Tp>
  _Tp std::arg (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::asin** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::asinh** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::atan** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::atanh** (const std::complex< _Tp > &)
- template<typename _Tp>
  constexpr std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **std::conj** (_Tp __x)
- template<typename _Tp>
  constexpr complex< _Tp > std::conj (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::cos (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::cosh (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::exp (const complex< _Tp > &)
- template<typename _Tp>
  _Tp std::fabs (const std::complex< _Tp > &__z)

- template<typename _Tp>
  constexpr __gnu_cxx::__promote< _Tp >::__type **std::imag** (_Tp)
- template<typename _Tp>
  constexpr _Tp **std::imag** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > std::log (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::log10 (const complex< _Tp > &)
- template<typename _Tp>
  constexpr __gnu_cxx::__promote< _Tp >::__type **std::norm** (_Tp __x)
- template<typename _Tp>
  _Tp constexpr std::norm (const complex< _Tp > &)
- template<typename _Tp>
  constexpr _Tp std::norm (const complex< _Tp > &__z)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x)
- template<typename _Tp, typename _CharT, class _Traits>
  basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const complex< _Tp > &__x)
- template ostream & **std::operator**<< (ostream &, const complex< double > &)
- template ostream & **std::operator**<< (ostream &, const complex< float > &)
- template ostream & **std::operator**<< (ostream &, const complex< long double > &)
- template wostream & **std::operator**<< (wostream &, const complex< double > &)
- template wostream & **std::operator**<< (wostream &, const complex< float > &)
- template wostream & **std::operator**<< (wostream &, const complex< long double > &)
- template<typename _Tp, typename _CharT, class _Traits>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, complex< _Tp > &__x)
- template istream & **std::operator**>> (istream &, complex< double > &)
- template istream & **std::operator**>> (istream &, complex< float > &)
- template istream & **std::operator**>> (istream &, complex< long double > &)
- template wistream & **std::operator**>> (wistream &, complex< double > &)
- template wistream & **std::operator**>> (wistream &, complex< float > &)
- template wistream & **std::operator**>> (wistream &, complex< long double > &)
- template<typename _Tp>
  complex< _Tp > std::polar (const _Tp &, const _Tp &=_Tp(0))
- template<typename _Tp>
  complex< _Tp > std::pow (const _Tp &, const complex< _Tp > &)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::pow** (const _Tp &__x, const std::complex< _Up > &__y)
- template<typename _Tp>
  complex< _Tp > std::pow (const complex< _Tp > &, const _Tp &)
- template<typename _Tp>
  complex< _Tp > std::pow (const complex< _Tp > &, const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::pow (const complex< _Tp > &, int)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::pow** (const std::complex< _Tp > &__x, const _Up &__y)

- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)
- template<typename _Tp>
  std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **std::proj** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **std::proj** (const std::complex< _Tp > &)
- template<typename _Tp>
  constexpr __gnu_cxx::__promote< _Tp >::__type **std::real** (_Tp __x)
- template<typename _Tp>
  constexpr _Tp **std::real** (const complex< _Tp > &__z)
- template<typename _Tp>
  complex< _Tp > std::sin (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::sinh (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::sqrt (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::tan (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::tanh (const complex< _Tp > &)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const _Tp &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator+ (const complex< _Tp > &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const _Tp &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator- (const complex< _Tp > &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator* (const _Tp &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator* (const complex< _Tp > &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator/ (const _Tp &__x, const complex< _Tp > &__y)
- template<typename _Tp>
  constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const _Tp &__y)

- template<typename _Tp>
  constexpr complex< _Tp > std::operator/ (const complex< _Tp > &__x, const complex< _Tp > &__y)

- template<typename _Tp>
  constexpr bool std::operator== (const complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp>
  constexpr bool std::operator== (const complex< _Tp > &__x, const complex< _Tp > &__y)

7.238.1 Detailed Description

This is a Standard C++ Library header.

7.239 tr1/complex File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_COMPLEX**

Functions

- template<typename _Tp>
  std::complex< _Tp > **std::tr1::acos** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::acosh** (const std::complex< _Tp > &)
- template<typename _Tp>
  _Tp std::tr1::arg (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::asin** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::asinh** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::atan** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::atanh** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< typename __gnu_cxx::__promote< _Tp >::__type > **std::tr1::conj** (_Tp __x)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::conj** (const std::complex< _Tp > &__z)
- template<typename _Tp>
  std::complex< _Tp > std::tr1::fabs (const std::complex< _Tp > &__z)
- template<typename _Tp>
  constexpr _Tp **std::tr1::imag** (const complex< _Tp > &__z)
- template<typename _Tp>
  _Tp constexpr std::tr1::norm (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > std::tr1::polar (const _Tp &, const _Tp &=_Tp(0))

- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::tr1::polar** (const _Tp &__rho, const _Up &__theta)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::pow** (const _Tp &__x, const std::complex< _Tp > &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::tr1::pow** (const _Tp &__x, const std::complex< _Up > &__y)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::pow** (const std::complex< _Tp > &__x, const _Tp &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > std::tr1::pow (const std::complex< _Tp > &__x, const _Up &__y)
- template<typename _Tp>
  std::complex< _Tp > **std::tr1::pow** (const std::complex< _Tp > &__x, const std::complex< _Tp > &__y)
- template<typename _Tp, typename _Up>
  std::complex< typename __gnu_cxx::__promote_2< _Tp, _Up >::__type > **std::tr1::pow** (const std::complex< _Tp > &__x, const std::complex< _Up > &__y)
- template<typename _Tp>
  constexpr _Tp **std::tr1::real** (const complex< _Tp > &__z)

7.239.1 Detailed Description

This is a TR1 C++ Library header.

7.240 complex.h File Reference

7.240.1 Detailed Description

This is a Standard C++ Library header.

7.241 Concepts

Here is a list of all documented concepts with brief descriptions:

std::assignable_from
[concept.assignable], concept assignable_from 733

std::common_reference_with
[concept.commonref], concept common_reference_with 733

std::common_with
[concept.common], concept common_with 734

std::constructible_from
[concept.constructible], concept constructible_from 734

std::convertible_to
[concept.convertible], concept convertible_to 734

std::copy_constructible
[concept.copyconstructible], concept copy_constructible 735

std::default_initializable
[concept.defaultinitializable], concept default_initializable 735

std::derived_from
[concept.derived], concept derived_from 735

std::destructible
[concept.destructible], concept destructible 735

std::equivalence_relation
[concept.equiv], concept equivalence_relation 736

std::indirectly_comparable
[alg.req.ind.cmp], concept `indirectly_comparable` 736

std::indirectly_copyable
[alg.req.ind.copy], concept `indirectly_copyable` 736

std::indirectly_movable
[alg.req.ind.move], concept `indirectly_movable` 736

std::indirectly_readable
Requirements for types that are readable by applying operator* 736

std::indirectly_swappable
[alg.req.ind.swap], concept `indirectly_swappable` 737

std::indirectly_writable
Requirements for writing a value into an iterator's referenced object 737

std::invocable
[concept.invocable], concept invocable 737

std::mergeable
[alg.req.mergeable], concept `mergeable` 738

std::move_constructible
[concept.moveconstructible], concept move_constructible 738

std::permutable
[alg.req.permutable], concept `permutable` 738

std::predicate
[concept.predicate], concept predicate 738

std::ranges::bidirectional_range
A range for which ranges::begin returns a bidirectional iterator 739

std::ranges::borrowed_range
[range.range] The borrowed_range concept 739

std::ranges::common_range
A range for which ranges::begin and ranges::end return the same type 739

std::ranges::contiguous_range
A range for which ranges::begin returns a contiguous iterator 739

std::ranges::forward_range
A range for which ranges::begin returns a forward iterator 739

std::ranges::input_range
A range for which ranges::begin returns an input iterator 740

std::ranges::output_range
A range for which ranges::begin returns an output iterator 740

std::ranges::random_access_range
A range for which ranges::begin returns a random access iterator 740

std::ranges::range
[range.range] The range concept 740

std::ranges::sized_range
[range.sized] The sized_range concept 741

std::ranges::view
[range.view] The ranges::view concept 741

std::ranges::viewable_range
A range which can be safely converted to a view 741

std::regular_invocable
[concept.regularinvocable], concept regular_invocable 741

std::relation
[concept.relation], concept relation 742

std::same_as
[concept.same], concept same_as 742

std::simd::__vec_builtin 742

std::simd::__vec_builtin_of 742

std::sortable
[alg.req.sortable], concept `sortable` 743

std::strict_weak_order
[concept.strictweakorder], concept strict_weak_order 743

std::uniform_random_bit_generator
Requirements for a uniform random bit generator 743

std::weakly_incrementable
Requirements on types that can be incremented with ++ 744

7.242 condition_variable File Reference

Classes

- class std::condition_variable
- class std::condition_variable_any

Namespaces

- namespace std

Enumerations

- enum class std::cv_status { **no_timeout** , **timeout** }

Functions

- void **std::notify_all_at_thread_exit** (condition_variable &, unique_lock< mutex >)

7.242.1 Detailed Description

This is a Standard C++ Library header.

7.243 csetjmp File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSETJMP**
- #define **setjmp**(env)

7.243.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `setjmp.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.244 csignal File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSIGNAL**

7.244.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `signal.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.245 cstdalign File Reference

Macros

- #define **_GLIBCXX_CSTDALIGN**

7.245.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

7.246 cstdarg File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTDARG**
- #define **va_end**(ap)

7.246.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `stdarg.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.247 tr1/cstdarg File Reference

Macros

- #define **_GLIBCXX_TR1_CSTDARG**

7.247.1 Detailed Description

This is a TR1 C++ Library header.

7.248 cstdbool File Reference

Macros

- #define **_GLIBCXX_CSTDBOOL**

7.248.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

7.249 tr1/cstdbool File Reference

Macros

- #define **_GLIBCXX_TR1_CSTDBOOL**

7.249.1 Detailed Description

This is a TR1 C++ Library header.

7.250 cstddef File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTDDEF**

7.250.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `stddef.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.251 cstdint File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTDINT**

7.251.1 Detailed Description

This is a Standard C++ Library header.

7.252 tr1/cstdint File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_CSTDINT**

7.252.1 Detailed Description

This is a TR1 C++ Library header.

7.253 cstdio File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTDIO**

7.253.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `stdio.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.254 tr1/cstdio File Reference

Macros

- #define **_GLIBCXX_TR1_CSTDIO**

7.254.1 Detailed Description

This is a TR1 C++ Library header.

7.255 cstdlib File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTDLIB**
- #define **_GLIBCXX_INCLUDE_NEXT_C_HEADERS**

Functions

- ldiv_t **std::div** (long __i, long __j) noexcept

7.255.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `stdlib.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.256 tr1/cstdlib File Reference

Macros

- #define **_GLIBCXX_TR1_CSTDLIB**

7.256.1 Detailed Description

This is a TR1 C++ Library header.

7.257 cstring File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CSTRING**

Functions

- void * **std::memchr** (void *__s, int __c, size_t __n)
- char * **std::strchr** (char *__s, int __n)
- char * **std::strpbrk** (char *__s1, const char *__s2)
- char * **std::strrchr** (char *__s, int __n)
- char * **std::strstr** (char *__s1, const char *__s2)

7.257.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `string.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.258 ctgmath File Reference

Macros

- #define **_GLIBCXX_CTGMATH**

7.258.1 Detailed Description

This is a Standard C++ Library header.

Since

C++11 (removed in C++20)

7.259 tr1/ctgmath File Reference

Macros

- #define **_GLIBCXX_TR1_CTGMATH**

7.259.1 Detailed Description

This is a TR1 C++ Library header.

7.260 ctime File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CTIME**

7.260.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `time.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.261 tr1/ctime File Reference

Macros

- #define **_GLIBCXX_TR1_CTIME**

7.261.1 Detailed Description

This is a TR1 C++ Library header.

7.262 cuchar File Reference

Macros

- #define **_GLIBCXX_CUCHAR**

7.262.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `uchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.263 cwchar File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CWCHAR**

Functions

- wchar_t * **std::wcschr** (wchar_t *__p, wchar_t __c)
- wchar_t * **std::wcspbrk** (wchar_t *__s1, const wchar_t *__s2)
- wchar_t * **std::wcsrchr** (wchar_t *__p, wchar_t __c)
- wchar_t * **std::wcsstr** (wchar_t *__s1, const wchar_t *__s2)
- wchar_t * **std::wmemchr** (wchar_t *__p, wchar_t __c, size_t __n)

7.263.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the *.h implementation files.
This is the C++ version of the Standard C Library header `wchar.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.264 tr1/cwchar File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_CWCHAR**

7.264.1 Detailed Description

This is a TR1 C++ Library header.

7.265 cwctype File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_CWCTYPE**

7.265.1 Detailed Description

This is a Standard C++ Library file. You should `#include` this file in your programs, rather than any of the `*`.h implementation files.
This is the C++ version of the Standard C Library header `wctype.h`, and its contents are (mostly) the same as that header, but are all contained in the namespace `std` (except for names which are defined as macros in C).

7.266 tr1/cwctype File Reference

Namespaces

- namespace std
- namespace std::tr1

Macros

- #define **_GLIBCXX_TR1_CWCTYPE**

7.266.1 Detailed Description

This is a TR1 C++ Library header.

7.267 assertions.h File Reference

Macros

- #define **__glibcxx_requires_non_empty_range**(_First, _Last)
- #define **__glibcxx_requires_nonempty**()
- #define **__glibcxx_requires_subscript**(_N)
- #define **_GLIBCXX_DEBUG_ASSERT**(_Condition)
- #define **_GLIBCXX_DEBUG_ONLY**(_Statement)
- #define **_GLIBCXX_DEBUG_PEDASSERT**(_Condition)

7.267.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.268 debug.h File Reference

Namespaces

- namespace __gnu_debug
- namespace std
- namespace std::__debug

Macros

- #define **__glibcxx_requires_can_decrement_range**(_First1, _Last1, _First2)
- #define **__glibcxx_requires_can_increment**(_First, _Size)
- #define **__glibcxx_requires_can_increment_range**(_First1, _Last1, _First2)
- #define **__glibcxx_requires_cond**(_Cond, _Msg)
- #define **__glibcxx_requires_heap**(_First, _Last)
- #define **__glibcxx_requires_heap_pred**(_First, _Last, _Pred)
- #define **__glibcxx_requires_irreflexive**(_First, _Last)
- #define **__glibcxx_requires_irreflexive2**(_First, _Last)
- #define **__glibcxx_requires_irreflexive_pred**(_First, _Last, _Pred)
- #define **__glibcxx_requires_irreflexive_pred2**(_First, _Last, _Pred)
- #define **__glibcxx_requires_partitioned_lower**(_First, _Last, _Value)
- #define **__glibcxx_requires_partitioned_lower_pred**(_First, _Last, _Value, _Pred)
- #define **__glibcxx_requires_partitioned_upper**(_First, _Last, _Value)
- #define **__glibcxx_requires_partitioned_upper_pred**(_First, _Last, _Value, _Pred)
- #define **__glibcxx_requires_sorted**(_First, _Last)
- #define **__glibcxx_requires_sorted_pred**(_First, _Last, _Pred)
- #define **__glibcxx_requires_sorted_set**(_First1, _Last1, _First2)
- #define **__glibcxx_requires_sorted_set_pred**(_First1, _Last1, _First2, _Pred)
- #define **__glibcxx_requires_string**(_String)
- #define **__glibcxx_requires_string_len**(_String, _Len)
- #define **__glibcxx_requires_valid_range**(_First, _Last)

7.268.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.269 formatter.h File Reference

Classes

- class __gnu_debug::type_info

Namespaces

- namespace __gnu_cxx
- namespace __gnu_debug
- namespace std

Macros

- #define **_GLIBCXX_TYPEID**(_Type)

Enumerations

- enum **_Debug_msg_id** {
  __msg_valid_range , **__msg_insert_singular** , **__msg_insert_different** , **__msg_erase_bad** ,
  __msg_erase_different , **__msg_subscript_oob** , **__msg_empty** , **__msg_unpartitioned** ,
  __msg_unpartitioned_pred , **__msg_unsorted** , **__msg_unsorted_pred** , **__msg_not_heap** ,
  __msg_not_heap_pred , **__msg_bad_bitset_write** , **__msg_bad_bitset_read** , **__msg_bad_bitset_flip** ,
  __msg_self_splice , **__msg_splice_alloc** , **__msg_splice_bad** , **__msg_splice_other** ,
  __msg_splice_overlap , **__msg_init_singular** , **__msg_init_copy_singular** , **__msg_init_const_singular** ,
  __msg_copy_singular , **__msg_bad_deref** , **__msg_bad_inc** , **__msg_bad_dec** ,
  __msg_iter_subscript_oob , **__msg_advance_oob** , **__msg_retreat_oob** , **__msg_iter_compare_bad** ,
  __msg_compare_different , **__msg_iter_order_bad** , **__msg_order_different** , **__msg_distance_bad** ,
  __msg_distance_different , **__msg_deref_istream** , **__msg_inc_istream** , **__msg_output_ostream** ,
  __msg_deref_istreambuf , **__msg_inc_istreambuf** , **__msg_insert_after_end** , **__msg_erase_after_bad** ,
  __msg_valid_range2 , **__msg_local_iter_compare_bad** , **__msg_non_empty_range** , **__msg_self_move↩
  _assign** ,
  __msg_bucket_index_oob , **__msg_valid_load_factor** , **__msg_equal_allocs** , **__msg_insert_range_↩
  from_self** ,
  __msg_irreflexive_ordering }

Functions

- template<typename _Iterator>
  constexpr bool **__gnu_debug::__check_singular** (_Iterator const &)

7.269.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.270 functions.h File Reference

Namespaces

- namespace __gnu_debug

Functions

- template<typename _ForwardIterator, typename _Tp>
  constexpr bool **__gnu_debug::__check_partitioned_lower** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _ForwardIterator, typename _Tp, typename _Pred>
  constexpr bool **__gnu_debug::__check_partitioned_lower** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)
- template<typename _ForwardIterator, typename _Tp>
  constexpr bool **__gnu_debug::__check_partitioned_upper** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value)
- template<typename _ForwardIterator, typename _Tp, typename _Pred>
  constexpr bool **__gnu_debug::__check_partitioned_upper** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__value, _Pred __pred)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__check_sorted** (const _InputIterator &__first, const _InputIterator &__last)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__gnu_debug::__check_sorted** (const _InputIterator &__first, const _InputIterator &__last, _↩
  Predicate __pred)

- template<typename _ForwardIterator, typename _Predicate>
  constexpr bool **__gnu_debug::__check_sorted_aux** (_ForwardIterator __first, _ForwardIterator __last, _↩ Predicate __pred, std::forward_iterator_tag)
- template<typename _ForwardIterator>
  constexpr bool **__gnu_debug::__check_sorted_aux** (_ForwardIterator __first, _ForwardIterator __last, std::forward_iterator_tag)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__gnu_debug::__check_sorted_aux** (const _InputIterator &, const _InputIterator &, _Predicate, std::input_iterator_tag)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__check_sorted_aux** (const _InputIterator &, const _InputIterator &, std::input_iterator_tag)
- template<typename _InputIterator1, typename _InputIterator2>
  constexpr bool **__gnu_debug::__check_sorted_set** (const _InputIterator1 &__first, const _InputIterator1 &__↩ last, const _InputIterator2 &)
- template<typename _InputIterator1, typename _InputIterator2, typename _Predicate>
  constexpr bool **__gnu_debug::__check_sorted_set** (const _InputIterator1 &__first, const _InputIterator1 &__↩ last, const _InputIterator2 &, _Predicate __pred)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__gnu_debug::__check_sorted_set_aux** (const _InputIterator &, const _InputIterator &, _↩ Predicate, std::__false_type)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__check_sorted_set_aux** (const _InputIterator &, const _InputIterator &, std::↩ __false_type)
- template<typename _InputIterator, typename _Predicate>
  constexpr bool **__gnu_debug::__check_sorted_set_aux** (const _InputIterator &__first, const _InputIterator &↩ __last, _Predicate __pred, std::__true_type)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__check_sorted_set_aux** (const _InputIterator &__first, const _InputIterator &↩ __last, std::__true_type)
- template<typename _InputIterator>
  constexpr _InputIterator **__gnu_debug::__check_valid_range** (const _InputIterator &__first, const _InputIterator &__last, const char *__file, unsigned int __line, const char *__function)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__gnu_debug::__foreign_iterator** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, _↩ InputIterator __other, _InputIterator __other_end)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator, typename _OtherSequence, typename _OtherCategory>
  bool __gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &, const _Safe_iterator< _OtherIterator, _OtherSequence, _OtherCategory > &)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__gnu_debug::__foreign_iterator_aux2** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _InputIterator &__other, const _InputIterator &__other_end)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _OtherIterator>
  bool __gnu_debug::__foreign_iterator_aux2 (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _Safe_iterator< _OtherIterator, _Sequence, _Category > &__other, const _Safe_iterator< _OtherIterator, _Sequence, _Category > &)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__gnu_debug::__foreign_iterator_aux3** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _InputIterator &, const _InputIterator &, std::__false_type)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _InputIterator>
  bool **__gnu_debug::__foreign_iterator_aux3** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const _InputIterator &__other, const _InputIterator &__other_end, std::__true_type)

- template<typename _Iterator, typename _Sequence, typename _Category>
  bool **__gnu_debug::__foreign_iterator_aux4** (const _Safe_iterator< _Iterator, _Sequence, _Category > &,...)
- template<typename _Iterator, typename _Sequence, typename _Category>
  bool **__gnu_debug::__foreign_iterator_aux4** (const _Safe_iterator< _Iterator, _Sequence, _Category > &__it, const typename _Sequence::value_type *__other)
- template<typename _Iterator>
  constexpr bool **__gnu_debug::__is_irreflexive** (_Iterator __it)
- template<typename _Iterator, typename _Pred>
  constexpr bool **__gnu_debug::__is_irreflexive_pred** (_Iterator __it, _Pred __pred)

7.270.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.271 helper_functions.h File Reference

Namespaces

- namespace __gnu_debug

Enumerations

- enum __gnu_debug::_Distance_precision {
  __dp_none , **__dp_equality** , **__dp_sign** , **__dp_sign_max_size** ,
  __dp_exact }

Functions

- template<typename _Iterator>
  constexpr _Iterator __gnu_debug::__base (_Iterator __it)
- template<typename _InputIterator, typename _Size>
  constexpr bool **__gnu_debug::__can_advance** (_InputIterator, _Size)
- template<typename _InputIterator, typename _Diff>
  constexpr bool **__gnu_debug::__can_advance** (_InputIterator, const std::pair< _Diff, _Distance_precision > &, int)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Size>
  constexpr bool **__gnu_debug::__can_advance** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, _Size)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Diff>
  constexpr bool **__gnu_debug::__can_advance** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const std::pair< _Diff, _Distance_precision > &, int)
- template<typename _Iterator>
  constexpr bool **__gnu_debug::__check_singular** (_Iterator const &)
- template<typename _Tp>
  constexpr bool __gnu_debug::__check_singular (_Tp *const &__ptr)
- bool **__gnu_debug::__check_singular_aux** (const class _Safe_iterator_base *)
- bool **__gnu_debug::__check_singular_aux** (const void *)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__gnu_debug::__get_distance** (_Iterator __lhs, _Iterator __↩ rhs)
- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type **__gnu_debug::__get_distance** (_Iterator __lhs, _Iterator __↩ rhs, std::input_iterator_tag)

- template<typename _Iterator>
  constexpr _Distance_traits< _Iterator >::__type __gnu_debug::__get_distance (_Iterator __lhs, _Iterator __rhs, std::random_access_iterator_tag)
- template<typename _Iterator>
  constexpr _Iterator **__gnu_debug::__unsafe** (_Iterator __it)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__valid_range** (_InputIterator __first, _InputIterator __last)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__valid_range** (_InputIterator __first, _InputIterator __last, typename _↩ Distance_traits< _InputIterator >::__type &__dist)
- template<typename _Iterator, typename _Sequence, typename _Category>
  constexpr bool **__gnu_debug::__valid_range** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _Iterator, _Sequence, _Category > &)
- template<typename _Iterator, typename _Sequence, typename _Category>
  constexpr bool __gnu_debug::__valid_range (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _Iterator, _Sequence, _Category > &, typename _Distance_traits< _Iterator >::__type &)
- template<typename _Iterator, typename _Sequence>
  bool **__gnu_debug::__valid_range** (const _Safe_local_iterator< _Iterator, _Sequence > &, const _Safe_local_iterator< _Iterator, _Sequence > &)
- template<typename _Iterator, typename _Sequence>
  bool **__gnu_debug::__valid_range** (const _Safe_local_iterator< _Iterator, _Sequence > &, const _Safe_local_iterator< _Iterator, _Sequence > &, typename _Distance_traits< _Iterator >::__type &)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__valid_range_aux** (_InputIterator __first, _InputIterator __last, std::input_iterator_tag)
- template<typename _InputIterator>
  constexpr bool **__gnu_debug::__valid_range_aux** (_InputIterator __first, _InputIterator __last, std::random_access_iterator_tag)

7.271.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.272 macros.h File Reference

Macros

- #define **__glibcxx_check_bucket_index**(_N)
- #define **__glibcxx_check_can_decrement_range**(_First1, _Last1, _First2)
- #define **__glibcxx_check_can_increment**(_First, _Size)
- #define **__glibcxx_check_can_increment_dist**(_First, _Dist, _Way)
- #define **__glibcxx_check_can_increment_range**(_First1, _Last1, _First2)
- #define **__glibcxx_check_equal_allocs**(_This, _Other)
- #define __glibcxx_check_erase(_Position)
- #define **__glibcxx_check_erase2**(_CPosition)
- #define __glibcxx_check_erase_after(_Position)
- #define __glibcxx_check_erase_range(_First, _Last)
- #define __glibcxx_check_erase_range_after(_First, _Last)
- #define **__glibcxx_check_heap**(_First, _Last)
- #define __glibcxx_check_heap_pred(_First, _Last, _Pred)
- #define __glibcxx_check_insert(_Position)
- #define __glibcxx_check_insert_after(_Position)
- #define __glibcxx_check_insert_range(_Position, _First, _Last, _Dist)
- #define __glibcxx_check_insert_range_after(_Position, _First, _Last, _Dist)
- #define **__glibcxx_check_irreflexive**(_First, _Last)

- #define **__glibcxx_check_irreflexive2**(_First, _Last)
- #define **__glibcxx_check_irreflexive_pred**(_First, _Last, _Pred)
- #define **__glibcxx_check_irreflexive_pred2**(_First, _Last, _Pred)
- #define **__glibcxx_check_max_load_factor**(_F)
- #define **__glibcxx_check_non_empty_range**(_First, _Last)
- #define **__glibcxx_check_nonempty**()
- #define __glibcxx_check_partitioned_lower(_First, _Last, _Value)
- #define __glibcxx_check_partitioned_lower_pred(_First, _Last, _Value, _Pred)
- #define **__glibcxx_check_partitioned_upper**(_First, _Last, _Value)
- #define __glibcxx_check_partitioned_upper_pred(_First, _Last, _Value, _Pred)
- #define **__glibcxx_check_sorted**(_First, _Last)
- #define __glibcxx_check_sorted_pred(_First, _Last, _Pred)
- #define **__glibcxx_check_sorted_set**(_First1, _Last1, _First2)
- #define **__glibcxx_check_sorted_set_pred**(_First1, _Last1, _First2, _Pred)
- #define **__glibcxx_check_subscript**(_N)
- #define **__glibcxx_check_valid_constructor_range**(_First, _Last)
- #define **__glibcxx_check_valid_range**(_First, _Last)
- #define **__glibcxx_check_valid_range2**(_First, _Last, _Dist)
- #define **__glibcxx_check_valid_range_at**(_First, _Last, _File, _Line, _Func)
- #define **_GLIBCXX_DEBUG_VERIFY**(_Cond, _ErrMsg)
- #define **_GLIBCXX_DEBUG_VERIFY_AT**(_Cond, _ErrMsg, _File, _Line)
- #define _GLIBCXX_DEBUG_VERIFY_AT_F(_Cond, _ErrMsg, _File, _Line, _Func)

7.272.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.272.2 Macro Definition Documentation

__glibcxx_check_erase

```
#define __glibcxx_check_erase(
              _Position)
```

Verify that we can erase the element referenced by the iterator _Position. We can erase the element if the _Position iterator is dereferenceable and references this sequence.

__glibcxx_check_erase_after

```
#define __glibcxx_check_erase_after(
              _Position)
```

Verify that we can erase the element after the iterator _Position. We can erase the element if the _Position iterator is before a dereferenceable one and references this sequence.

__glibcxx_check_erase_range

```
#define __glibcxx_check_erase_range(
              _First,
              _Last)
```

Verify that we can erase the elements in the iterator range [_First, _Last). We can erase the elements if [_First, _Last) is a valid iterator range within this sequence.

__glibcxx_check_erase_range_after

```
#define __glibcxx_check_erase_range_after(
              _First,
              _Last)
```

Verify that we can erase the elements in the iterator range (_First, _Last). We can erase the elements if (_First, _Last) is a valid iterator range within this sequence.

__glibcxx_check_heap_pred

```
#define __glibcxx_check_heap_pred(
              _First,
              _Last,
              _Pred)
```

Verify that the iterator range [_First, _Last) is a heap w.r.t. the predicate _Pred.

__glibcxx_check_insert

```
#define __glibcxx_check_insert(
              _Position)
```

Verify that we can insert into ∗this with the iterator _Position. Insertion into a container at a specific position requires that the iterator be nonsingular, either dereferenceable or past-the-end, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a_Safe_sequence and the iterator is a _Safe_iterator.

__glibcxx_check_insert_after

```
#define __glibcxx_check_insert_after(
              _Position)
```

Verify that we can insert into ∗this after the iterator _Position. Insertion into a container after a specific position requires that the iterator be nonsingular, either dereferenceable or before-begin, and that it reference the sequence we are inserting into. Note that this macro is only valid when the container is a_Safe_sequence and the iterator is a _Safe_↩ iterator.

__glibcxx_check_insert_range

```
#define __glibcxx_check_insert_range(
              _Position,
              _First,
              _Last,
              _Dist)
```

Verify that we can insert the values in the iterator range [_First, _Last) into ∗this with the iterator _Position. Insertion into a container at a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range [_First, _Last) is a valid (possibly empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a _Safe_sequence and the _Position iterator is a _Safe_iterator.

__glibcxx_check_insert_range_after

```
#define __glibcxx_check_insert_range_after(
              _Position,
              _First,
              _Last,
              _Dist)
```

Verify that we can insert the values in the iterator range [_First, _Last) into ∗this after the iterator _Position. Insertion into a container after a specific position requires that the iterator be nonsingular (i.e., either dereferenceable or past-the-end), that it reference the sequence we are inserting into, and that the iterator range [_First, _Last) is a valid (possibly

empty) range which does not reference the sequence we are inserting into. Note that this macro is only valid when the container is a _Safe_sequence and the _Position iterator is a _Safe_iterator.

__glibcxx_check_partitioned_lower

```
#define __glibcxx_check_partitioned_lower(
              _First,
              _Last,
              _Value)
```

Verify that the iterator range [_First, _Last) is partitioned w.r.t. the value _Value.

__glibcxx_check_partitioned_lower_pred

```
#define __glibcxx_check_partitioned_lower_pred(
              _First,
              _Last,
              _Value,
              _Pred)
```

Verify that the iterator range [_First, _Last) is partitioned w.r.t. the value _Value and predicate _Pred.

__glibcxx_check_partitioned_upper_pred

```
#define __glibcxx_check_partitioned_upper_pred(
              _First,
              _Last,
              _Value,
              _Pred)
```

Verify that the iterator range [_First, _Last) is partitioned w.r.t. the value _Value and predicate _Pred.

__glibcxx_check_sorted_pred

```
#define __glibcxx_check_sorted_pred(
              _First,
              _Last,
              _Pred)
```

Verify that the iterator range [_First, _Last) is sorted by the predicate _Pred.

_GLIBCXX_DEBUG_VERIFY_AT_F

```
#define _GLIBCXX_DEBUG_VERIFY_AT_F(
              _Cond,
              _ErrMsg,
              _File,
              _Line,
              _Func)
```

Macros used by the implementation to verify certain properties. These macros may only be used directly by the debug wrappers. Note that these are macros (instead of the more obviously *correct* choice of making them functions) because we need line and file information at the call site, to minimize the distance between the user error and where the error is reported.

7.273 map.h File Reference

Classes

- class std::__debug::map< _Key, _Tp, _Compare, _Allocator >

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::__debug::map (_InputIterator, _InputIterator, _Allocator) -> map< __iter_key_t< _InputIterator >, __iter↩ _val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter↩ _to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::map (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> map< _Key, _Tp, less< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::map (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> map< _Key, _Tp, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< pair< const _Key, _Tp > > **std::__debug::operator<=>** (const map< _Key, _Tp, _Compare, _Alloc > &__lhs, const map< _Key, _Tp, _Compare, _Alloc > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  bool **std::__debug::operator==** (const map< _Key, _Tp, _Compare, _Allocator > &__lhs, const map< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  void **std::__debug::swap** (map< _Key, _Tp, _Compare, _Allocator > &__lhs, map< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)

7.273.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.274 multimap.h File Reference

Classes

- class std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::__debug::multimap (_InputIterator, _InputIterator, _Allocator) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, less< __iter_key_t< _InputIterator > >, _Allocator >

- template<typename _InputIterator, typename _Compare = less<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::multimap (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> multimap< _Key, _Tp, less< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::multimap (initializer_list< pair< _Key, _Tp > >, _Compare=_Compare(), _Allocator=_Allocator()) -> multimap< _Key, _Tp, _Compare, _Allocator >
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< pair< const _Key, _Tp > > **std::__debug::operator<=>** (const multimap< _Key, _Tp, _Compare, _Alloc > &__lhs, const multimap< _Key, _Tp, _Compare, _Alloc > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  bool **std::__debug::operator==** (const multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, const multimap< _Key, _Tp, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Tp, typename _Compare, typename _Allocator>
  void **std::__debug::swap** (multimap< _Key, _Tp, _Compare, _Allocator > &__lhs, multimap< _Key, _Tp, _Compare, _Allocator > &__rhs) noexcept(/*conditional */)

7.274.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.275 multiset.h File Reference

Classes

- class std::__debug::multiset< _Key, _Compare, _Allocator >

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::multiset (_InputIterator, _InputIterator, _Allocator) -> multiset< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::multiset (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::multiset (initializer_list< _Key >, _Allocator) -> multiset< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::multiset (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> multiset< _Key, _Compare, _Allocator >

- template<typename _Key, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< _Key > **std::__debug::operator<=>** (const multiset< _Key, _Compare, _Alloc > &__lhs, const multiset< _Key, _Compare, _Alloc > &__rhs)
- template<typename _Key, typename _Compare, typename _Allocator>
  bool **std::__debug::operator==** (const multiset< _Key, _Compare, _Allocator > &__lhs, const multiset< _Key, _Compare, _Allocator > &__rhs)
- template<typename _Key, typename _Compare, typename _Allocator>
  void **std::__debug::swap** (multiset< _Key, _Compare, _Allocator > &__x, multiset< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */)

7.275.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.276 safe_base.h File Reference

Classes

- class __gnu_debug::_Safe_iterator_base
- class __gnu_debug::_Safe_sequence_base

Namespaces

- namespace __gnu_debug

Functions

- bool __gnu_debug::__check_singular_aux (const _Safe_iterator_base *__x)

7.276.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.277 safe_container.h File Reference

Classes

- class __gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware >

Namespaces

- namespace __gnu_debug

7.277.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.278 safe_iterator.h File Reference

Classes

- struct __gnu_debug::_BeforeBeginHelper< _Sequence >
- class __gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >
- struct __gnu_debug::_Sequence_traits< _Sequence >

Namespaces

- namespace __gnu_debug

Macros

- #define **_GLIBCXX20_CONSTEXPR_NON_LITERAL_SCOPE_BEGIN**
- #define **_GLIBCXX20_CONSTEXPR_NON_LITERAL_SCOPE_END**
- #define **_GLIBCXX_DEBUG_VERIFY_DIST_OPERANDS**(_Lhs, _Rhs)
- #define **_GLIBCXX_DEBUG_VERIFY_EQ_OPERANDS**(_Lhs, _Rhs)
- #define **_GLIBCXX_DEBUG_VERIFY_OPERANDS**(_Lhs, _Rhs, _BadMsgId, _DiffMsgId)
- #define **_GLIBCXX_DEBUG_VERIFY_REL_OPERANDS**(_Lhs, _Rhs)

Functions

- template<typename _Iterator, typename _Sequence>
  constexpr _Iterator **__gnu_debug::__base** (const _Safe_iterator< _Iterator, _Sequence, std::random_access_iterator_tag > &__it)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Size>
  constexpr bool **__gnu_debug::__can_advance** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, _Size)
- template<typename _Iterator, typename _Sequence, typename _Category, typename _Diff>
  constexpr bool **__gnu_debug::__can_advance** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const std::pair< _Diff, _Distance_precision > &, int)
- template<typename _Iterator, typename _Sequence>
  _Iterator **__gnu_debug::__unsafe** (const _Safe_iterator< _Iterator, _Sequence > &__it)
- template<typename _Iterator, typename _Sequence, typename _Category>
  constexpr bool **__gnu_debug::__valid_range** (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _Iterator, _Sequence, _Category > &)
- template<typename _Iterator, typename _Sequence, typename _Category>
  constexpr bool __gnu_debug::__valid_range (const _Safe_iterator< _Iterator, _Sequence, _Category > &, const _Safe_iterator< _Iterator, _Sequence, _Category > &, typename _Distance_traits< _Iterator >::__type &)

7.278.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.279 safe_iterator.tcc File Reference

Namespaces

- namespace __gnu_debug
- namespace std

Functions

- template<bool _IsMove, typename _II, typename _Ite, typename _Seq, typename _Cat>
  constexpr __gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **std::__copy_move_a** (_II, _II, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &)
- template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>
  constexpr ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > **std::__copy_move_a** (const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > &)

- template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _Ol>
  constexpr _Ol **std::__copy_move_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, _Ol)
- template<bool _IsMove, typename _Il, typename _Ite, typename _Seq, typename _Cat>
  constexpr __gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **std::__copy_move_backward_a** (_Il, _Il, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &)
- template<bool _IsMove, typename _IIte, typename _ISeq, typename _ICat, typename _OIte, typename _OSeq, typename _OCat>
  constexpr ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > **std::__copy_move_backward_a** (const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _ICat > &, const ::__gnu_debug::_Safe_iterator< _IIte, _ISeq, _↩ ICat > &, const ::__gnu_debug::_Safe_iterator< _OIte, _OSeq, _OCat > &)
- template<bool _IsMove, typename _Ite, typename _Seq, typename _Cat, typename _Ol>
  constexpr _Ol **std::__copy_move_backward_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, _Ol)
- template<typename _II1, typename _II2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__equal_aux** (_II1, _II1, const ::__gnu_debug::_Safe_iterator< _II2, _Seq2, _Cat2 > &)
- template<typename _II1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **std::__equal_aux** (const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, _II2)
- template<typename _II1, typename _Seq1, typename _Cat1, typename _II2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__equal_aux** (const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II1, _Seq1, _Cat1 > &, const ::__gnu_debug::_Safe_iterator< _II2, _Seq2, _Cat2 > &)
- template<typename _Ite, typename _Seq, typename _Cat, typename _Tp>
  constexpr void **std::__fill_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &, const _Tp &)
- template<typename _Ite, typename _Seq, typename _Cat, typename _Size, typename _Tp>
  constexpr ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > **std::__fill_n_a** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, _Cat > &__first, _Size __n, const _Tp &__value, std::input_iterator_tag)
- template<typename _II1, typename _Ite2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__lexicographical_compare_aux** (_II1 __first1, _II1 __last1, const ::__gnu_debug::_Safe_iterator< _Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::_Safe_iterator< _Ite2, _Seq2, _Cat2 > &__last2)
- template<typename _Ite1, typename _Seq1, typename _Cat1, typename _II2>
  constexpr bool **std::__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, ↩ _Cat1 > &__first1, const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, _II2 __first2, _II2 __last2)
- template<typename _Ite1, typename _Seq1, typename _Cat1, typename _Ite2, typename _Seq2, typename _Cat2>
  constexpr bool **std::__lexicographical_compare_aux** (const ::__gnu_debug::_Safe_iterator< _Ite1, _↩ Seq1, _Cat1 > &__first1, const ::__gnu_debug::_Safe_iterator< _Ite1, _Seq1, _Cat1 > &__last1, const ::__gnu_debug::_Safe_iterator< _Ite2, _Seq2, _Cat2 > &__first2, const ::__gnu_debug::_Safe_iterator< ↩ _Ite2, _Seq2, _Cat2 > &__last2)
- template<typename _Ite, typename _Seq>
  constexpr decltype(std::__niter_base(std::declval< _Ite >())) **std::__niter_base** (const ::__gnu_debug::_Safe_iterator< _Ite, _Seq, std::random_access_iterator_tag > &__it) noexcept(std::is_nothrow_copy_constructible< _Ite >::↩ value)

7.279.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.280 safe_local_iterator.h File Reference

Classes

- class __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >

Namespaces

- namespace __gnu_debug

Macros

- #define **_GLIBCXX_DEBUG_VERIFY_OPERANDS**(_Lhs, _Rhs)

Functions

- template<typename _Iterator, typename _UContainer>
  _Iterator **__gnu_debug::__unsafe** (const _Safe_local_iterator< _Iterator, _UContainer > &__it)
- template<typename _Iterator, typename _UContainer>
  bool **__gnu_debug::__valid_range** (const _Safe_local_iterator< _Iterator, _UContainer > &__first, const _Safe_local_iterator< _Iterator, _UContainer > &__last)
- template<typename _Iterator, typename _UContainer>
  bool __gnu_debug::__valid_range (const _Safe_local_iterator< _Iterator, _UContainer > &__first, const _Safe_local_iterator< _Iterator, _UContainer > &__last, typename _Distance_traits< _Iterator >::__type &__↩
  dist_info)

7.280.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.281 safe_local_iterator.tcc File Reference

Namespaces

- namespace __gnu_debug

7.281.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.282 safe_sequence.h File Reference

Classes

- class __gnu_debug::_After_nth_from< _Iterator >
- class __gnu_debug::_Equal_to< _Type >
- class __gnu_debug::_Not_equal_to< _Type >
- class __gnu_debug::_Safe_node_sequence< _Sequence >
- class __gnu_debug::_Safe_sequence< _Sequence >

Namespaces

- namespace __gnu_debug

7.282.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.283 safe_sequence.tcc File Reference

Namespaces

- namespace __gnu_debug

7.283.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.284 safe_unordered_base.h File Reference

Classes

- class __gnu_debug::_Safe_local_iterator_base
- class __gnu_debug::_Safe_unordered_container_base

Namespaces

- namespace __gnu_debug

7.284.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.285 safe_unordered_container.h File Reference

Classes

- class __gnu_debug::_Safe_unordered_container< _Container >

Namespaces

- namespace __gnu_debug

7.285.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.286 safe_unordered_container.tcc File Reference

Namespaces

- namespace __gnu_debug

7.286.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.287 set.h File Reference

Classes

- class std::__debug::set< _Key, _Compare, _Allocator >

Namespaces

- namespace std
- namespace std::__debug

Functions

- template<typename _Key, typename _Compare, typename _Alloc>
  __detail::__synth3way_t< _Key > **std::__debug::operator<=>** (const set< _Key, _Compare, _Alloc > &__lhs, const set< _Key, _Compare, _Alloc > &__rhs)
- template<typename _Key, typename _Compare, typename _Allocator>
  bool **std::__debug::operator==** (const set< _Key, _Compare, _Allocator > &__lhs, const set< _Key, _Compare, _Allocator > &__rhs)
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _Require↩ Allocator<_Allocator>>
  std::__debug::set (_InputIterator, _InputIterator, _Allocator) -> set< typename iterator_traits< _InputIterator >::value_type, less< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Compare = less<typename iterator_traits<_InputIterator>::value_type>, typename _↩ Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::set (_InputIterator, _InputIterator, _Compare=_Compare(), _Allocator=_Allocator()) -> set< typename iterator_traits< _InputIterator >::value_type, _Compare, _Allocator >
- template<typename _Key, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::set (initializer_list< _Key >, _Allocator) -> set< _Key, less< _Key >, _Allocator >
- template<typename _Key, typename _Compare = less<_Key>, typename _Allocator = allocator<_Key>, typename = _RequireNot↩ Allocator<_Compare>, typename = _RequireAllocator<_Allocator>>
  std::__debug::set (initializer_list< _Key >, _Compare=_Compare(), _Allocator=_Allocator()) -> set< _Key, _↩ Compare, _Allocator >
- template<typename _Key, typename _Compare, typename _Allocator>
  void **std::__debug::swap** (set< _Key, _Compare, _Allocator > &__x, set< _Key, _Compare, _Allocator > &__y) noexcept(/*conditional */)

7.287.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.288 debugging File Reference

7.288.1 Detailed Description

This is a Standard C++ Library header.

7.289 decimal File Reference

Classes

- class std::decimal::decimal128
- class std::decimal::decimal32
- class std::decimal::decimal64

Namespaces

- namespace std
- namespace std::decimal

Macros

- #define **_DECLARE_DECIMAL128_COMPOUND_ASSIGNMENT**(_Op)
- #define **_DECLARE_DECIMAL32_COMPOUND_ASSIGNMENT**(_Op)
- #define **_DECLARE_DECIMAL64_COMPOUND_ASSIGNMENT**(_Op)

- #define **_DECLARE_DECIMAL_BINARY_OP_WITH_DEC**(_Op, _T1, _T2, _T3)
- #define **_DECLARE_DECIMAL_BINARY_OP_WITH_INT**(_Op, _Tp)
- #define **_DECLARE_DECIMAL_COMPARISON**(_Op, _Tp)
- #define **_GLIBCXX_DECIMAL**
- #define **_GLIBCXX_USE_DECIMAL_**

Functions

- double **std::decimal::decimal128_to_double** (decimal128 __d)
- float **std::decimal::decimal128_to_float** (decimal128 __d)
- long double **std::decimal::decimal128_to_long_double** (decimal128 __d)
- long long **std::decimal::decimal128_to_long_long** (decimal128 __d)
- double **std::decimal::decimal32_to_double** (decimal32 __d)
- float **std::decimal::decimal32_to_float** (decimal32 __d)
- long double **std::decimal::decimal32_to_long_double** (decimal32 __d)
- long long std::decimal::decimal32_to_long_long (decimal32 __d)
- double **std::decimal::decimal64_to_double** (decimal64 __d)
- float **std::decimal::decimal64_to_float** (decimal64 __d)
- long double **std::decimal::decimal64_to_long_double** (decimal64 __d)
- long long **std::decimal::decimal64_to_long_long** (decimal64 __d)
- double **std::decimal::decimal_to_double** (decimal128 __d)
- double **std::decimal::decimal_to_double** (decimal32 __d)
- double **std::decimal::decimal_to_double** (decimal64 __d)
- float **std::decimal::decimal_to_float** (decimal128 __d)
- float **std::decimal::decimal_to_float** (decimal32 __d)
- float **std::decimal::decimal_to_float** (decimal64 __d)
- long double **std::decimal::decimal_to_long_double** (decimal128 __d)
- long double **std::decimal::decimal_to_long_double** (decimal32 __d)
- long double **std::decimal::decimal_to_long_double** (decimal64 __d)
- long long **std::decimal::decimal_to_long_long** (decimal128 __d)
- long long **std::decimal::decimal_to_long_long** (decimal32 __d)
- long long **std::decimal::decimal_to_long_long** (decimal64 __d)
- static decimal128 **std::decimal::make_decimal128** (long long __coeff, int __exp)
- static decimal128 **std::decimal::make_decimal128** (unsigned long long __coeff, int __exp)
- static decimal32 **std::decimal::make_decimal32** (long long __coeff, int __exp)
- static decimal32 **std::decimal::make_decimal32** (unsigned long long __coeff, int __exp)
- static decimal64 **std::decimal::make_decimal64** (long long __coeff, int __exp)
- static decimal64 **std::decimal::make_decimal64** (unsigned long long __coeff, int __exp)
- bool **std::decimal::operator!=** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator!=** (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, int __rhs)

- bool **std::decimal::operator!=** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator!=** (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator!=** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator!=** (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator!=** (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator!=** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator!=** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator*** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator*** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator*** (decimal32 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator*** (decimal64 __lhs, decimal128 __rhs)

- decimal64 **std::decimal::operator*** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, long __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator*** (decimal64 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator*** (int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (unsigned int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (unsigned int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (unsigned int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (unsigned long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (unsigned long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (unsigned long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator*** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator*** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator*** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator+** (decimal128 __rhs)
- decimal128 **std::decimal::operator+** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **std::decimal::operator+** (decimal32 __rhs)
- decimal128 **std::decimal::operator+** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, long __rhs)

- decimal64 **std::decimal::operator+** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **std::decimal::operator+** (decimal64 __rhs)
- decimal128 **std::decimal::operator+** (int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (unsigned int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (unsigned int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (unsigned int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (unsigned long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (unsigned long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (unsigned long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator+** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator+** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator+** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator-** (decimal128 __rhs)
- decimal128 **std::decimal::operator-** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __lhs, unsigned long long __rhs)
- decimal32 **std::decimal::operator-** (decimal32 __rhs)
- decimal128 **std::decimal::operator-** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, long __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __lhs, unsigned long __rhs)

- decimal64 **std::decimal::operator-** (decimal64 __lhs, unsigned long long __rhs)
- decimal64 **std::decimal::operator-** (decimal64 __rhs)
- decimal128 **std::decimal::operator-** (int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (unsigned int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (unsigned int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (unsigned int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (unsigned long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (unsigned long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (unsigned long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator-** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator-** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator-** (unsigned long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, decimal128 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, decimal32 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, int __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, long __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, long long __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, unsigned int __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, unsigned long __rhs)
- decimal128 **std::decimal::operator/** (decimal128 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator/** (decimal32 __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (decimal32 __lhs, decimal64 __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, int __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, long __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, long long __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, unsigned int __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, unsigned long __rhs)
- decimal32 **std::decimal::operator/** (decimal32 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator/** (decimal64 __lhs, decimal128 __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, decimal64 __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, int __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, long __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, long long __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, unsigned int __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, unsigned long __rhs)
- decimal64 **std::decimal::operator/** (decimal64 __lhs, unsigned long long __rhs)
- decimal128 **std::decimal::operator/** (int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (long __lhs, decimal128 __rhs)

- decimal32 **std::decimal::operator/** (long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (long long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (unsigned int __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (unsigned int __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (unsigned int __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (unsigned long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (unsigned long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (unsigned long __lhs, decimal64 __rhs)
- decimal128 **std::decimal::operator/** (unsigned long long __lhs, decimal128 __rhs)
- decimal32 **std::decimal::operator/** (unsigned long long __lhs, decimal32 __rhs)
- decimal64 **std::decimal::operator/** (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator<** (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator<** (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator<** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator<** (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (unsigned int __lhs, decimal128 __rhs)

- bool **std::decimal::operator<** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator<** (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator<** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator<** (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator==** (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator==** (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator==** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator==** (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator==** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator==** (unsigned long long __lhs, decimal128 __rhs)

- bool **std::decimal::operator==** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator==** (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, int __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator>** (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator>** (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator>** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator>** (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator>** (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator>** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator>** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator>** (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator>** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator>** (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator>** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator>** (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator>** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator>** (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator>** (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>** (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>** (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator>** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator>** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator>** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>** (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>** (unsigned long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, int __rhs)

- bool **std::decimal::operator>=** (decimal128 __lhs, long __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, long long __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, unsigned int __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, unsigned long __rhs)
- bool **std::decimal::operator>=** (decimal128 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, int __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, long __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, long long __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, unsigned int __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, unsigned long __rhs)
- bool **std::decimal::operator>=** (decimal32 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, int __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, long __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, long long __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, unsigned int __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, unsigned long __rhs)
- bool **std::decimal::operator>=** (decimal64 __lhs, unsigned long long __rhs)
- bool **std::decimal::operator>=** (int __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (int __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (int __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (long long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (unsigned int __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (unsigned int __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (unsigned int __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (unsigned long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (unsigned long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (unsigned long __lhs, decimal64 __rhs)
- bool **std::decimal::operator>=** (unsigned long long __lhs, decimal128 __rhs)
- bool **std::decimal::operator>=** (unsigned long long __lhs, decimal32 __rhs)
- bool **std::decimal::operator>=** (unsigned long long __lhs, decimal64 __rhs)

7.289.1 Detailed Description

This is a Standard C++ Library header.

7.290 debug/deque File Reference

Classes

- class std::__debug::deque< _Tp, _Allocator >

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_DEQUE**

Functions

- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::deque (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> deque< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  std::__debug::deque (size_t, _Tp, _Allocator=_Allocator()) -> deque< _Tp, _Allocator >
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **std::__debug::operator<=>** (const deque< _Tp, _Alloc > &__x, const deque< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool **std::__debug::operator==** (const deque< _Tp, _Alloc > &__lhs, const deque< _Tp, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc>
  void **std::__debug::swap** (deque< _Tp, _Alloc > &__lhs, deque< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)

7.290.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.291 deque File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Tp>
  using **std::pmr::deque**

7.291.1 Detailed Description

This is a Standard C++ Library header.

7.292 experimental/deque File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_DEQUE**

Typedefs

- template<typename _Tp>
  using **std::experimental::fundamentals_v2::pmr::deque**

Functions

- template<typename _Tp, typename _Alloc, typename _Up>
  void **std::experimental::erase** (deque< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (deque< _Tp, _Alloc > &__cont, _Predicate __pred)

7.292.1 Detailed Description

This is a TS C++ Library header.

7.293 expected File Reference

7.293.1 Detailed Description

This is a Standard C++ Library header.

7.294 lfts_config.h File Reference

7.294.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly.

7.295 ext/numeric_traits.h File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **__glibcxx_digits10**(_Tp)
- #define **__glibcxx_floating**(_Tp, _Fval, _Dval, _LDval)
- #define **__glibcxx_max_digits10**(_Tp)
- #define **__glibcxx_max_exponent10**(_Tp)
- #define **_GLIBCXX_INT_N_TRAITS**(T, WIDTH)

Typedefs

- template<typename _Tp>
  using __gnu_cxx::__int_traits

Variables

- template<typename _Value>
  const int **__gnu_cxx::__numeric_traits_floating**< **_Value** >**::__digits10**
- template<typename _Value>
  const bool **__gnu_cxx::__numeric_traits_floating**< **_Value** >**::__is_signed**
- template<typename _Value>
  const int **__gnu_cxx::__numeric_traits_floating**< **_Value** >**::__max_digits10**

- template<typename _Value>
  const int **__gnu_cxx::__numeric_traits_floating**< **_Value** >**::__max_exponent10**
- template<typename _Value>
  const int **__gnu_cxx::__numeric_traits_integer**< **_Value** >**::__digits**
- template<typename _Value>
  const bool **__gnu_cxx::__numeric_traits_integer**< **_Value** >**::__is_signed**
- template<typename _Value>
  const _Value **__gnu_cxx::__numeric_traits_integer**< **_Value** >**::__max**
- template<typename _Value>
  const _Value **__gnu_cxx::__numeric_traits_integer**< **_Value** >**::__min**

7.295.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.296 propagate_const File Reference

Classes

- class std::experimental::fundamentals_v2::propagate_const< _Tp >

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_PROPAGATE_CONST**

Typedefs

- template<typename _Tp>
  using **std::experimental::__propagate_const_elem_type**

Functions

- template<typename _Tp>
  constexpr const _Tp & **std::experimental::get_underlying** (const propagate_const< _Tp > &__pt) noexcept
- template<typename _Tp>
  constexpr _Tp & **std::experimental::get_underlying** (propagate_const< _Tp > &__pt) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr bool **std::experimental::operator!=** (const propagate_const< _Tp > &__pt, nullptr_t)
- template<typename _Tp>
  constexpr bool **std::experimental::operator!=** (nullptr_t, const propagate_const< _Tp > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**< (const _Tp &__t, const propagate_const< _Up > &__pu)

- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator<** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator<** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator<=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator<=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator<=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr bool **std::experimental::operator==** (const propagate_const< _Tp > &__pt, nullptr_t)
- template<typename _Tp>
  constexpr bool **std::experimental::operator==** (nullptr_t, const propagate_const< _Tp > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>=** (const _Tp &__t, const propagate_const< _Up > &__pu)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>=** (const propagate_const< _Tp > &__pt, const _Up &__u)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>=** (const propagate_const< _Tp > &__pt, const propagate_const< _Up > &__pu)
- template<typename _Tp>
  constexpr enable_if_t< __is_swappable< _Tp >::value, void > **std::experimental::swap** (propagate_const< _Tp > &__pt, propagate_const< _Tp > &__pt2) noexcept(__is_nothrow_swappable< _Tp >::value)

7.296.1 Detailed Description

This is a TS C++ Library header.

7.297 simd File Reference

Macros

- #define **__cpp_lib_experimental_parallel_simd**
- #define **_GLIBCXX_EXPERIMENTAL_SIMD**

7.297.1 Detailed Description

This is a TS C++ Library header.

7.298 aligned_buffer.h File Reference

Namespaces

- namespace __gnu_cxx

7.298.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.299 atomicity.h File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_GLIBCXX_READ_MEM_BARRIER**
- #define **_GLIBCXX_UNSIGNED_ATOMIC_WORD**
- #define **_GLIBCXX_WRITE_MEM_BARRIER**

Functions

- void **__gnu_cxx::__atomic_add** (volatile _Atomic_word *__mem, int __val)
- void **__gnu_cxx::__atomic_add_dispatch** (_Atomic_word *__mem, int __val)
- void **__gnu_cxx::__atomic_add_single** (_Atomic_word *__mem, int __val)
- _Atomic_word **__gnu_cxx::__exchange_and_add** (volatile _Atomic_word *__mem, int __val)
- _Atomic_word **__gnu_cxx::__exchange_and_add_dispatch** (_Atomic_word *__mem, int __val)
- _Atomic_word **__gnu_cxx::__exchange_and_add_single** (_Atomic_word *__mem, int __val)
- bool **__gnu_cxx::__is_single_threaded** () noexcept

7.299.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.300 bitmap_allocator.h File Reference

Classes

- class __gnu_cxx::__detail::__mini_vector< _Tp >
- class __gnu_cxx::__detail::_Bitmap_counter< _Tp >
- class __gnu_cxx::__detail::_Ffit_finder< _Tp >
- class __gnu_cxx::bitmap_allocator< _Tp >
- class __gnu_cxx::free_list

Namespaces

- namespace __gnu_cxx
- namespace __gnu_cxx::__detail

Macros

- #define _BALLOC_ALIGN_BYTES

Enumerations

- enum { **bits_per_byte** , **bits_per_block** }

Functions

- void __gnu_cxx::__detail::__bit_allocate (std::size_t *__pbmap, std::size_t __pos) throw ()
- void __gnu_cxx::__detail::__bit_free (std::size_t *__pbmap, std::size_t __pos) throw ()
- template<typename _ForwardIterator, typename _Tp, typename _Compare>
  _ForwardIterator **__gnu_cxx::__detail::__lower_bound** (_ForwardIterator __first, _ForwardIterator __last, const _Tp &__val, _Compare __comp)
- template<typename _AddrPair>
  std::size_t __gnu_cxx::__detail::__num_bitmaps (_AddrPair __ap)
- template<typename _AddrPair>
  std::size_t __gnu_cxx::__detail::__num_blocks (_AddrPair __ap)
- std::size_t __gnu_cxx::_Bit_scan_forward (std::size_t __num)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator==** (const bitmap_allocator< _Tp1 > &, const bitmap_allocator< _Tp2 > &) throw ()

Variables

- template<typename _Tp>
  std::size_t **__gnu_cxx::bitmap_allocator< _Tp >::_S_block_size**
- template<typename _Tp>
  bitmap_allocator< _Tp >::_BPVector::size_type **__gnu_cxx::bitmap_allocator< _Tp >::_S_last_dealloc_index**
- template<typename _Tp>
  bitmap_allocator< _Tp >::_BPVector **__gnu_cxx::bitmap_allocator< _Tp >::_S_mem_blocks**
- template<typename _Tp>
  bitmap_allocator< _Tp >::__mutex_type **__gnu_cxx::bitmap_allocator< _Tp >::_S_mut**

7.300.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.300.2 Macro Definition Documentation

_BALLOC_ALIGN_BYTES

```
#define _BALLOC_ALIGN_BYTES
```

The constant in the expression below is the alignment required in bytes.

7.301 cast.h File Reference

Classes

- struct __gnu_cxx::_Caster< _ToType >

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _ToType, typename _FromType>
  _ToType **__gnu_cxx::__const_pointer_cast** (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__gnu_cxx::__const_pointer_cast** (const _FromType &__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__gnu_cxx::__dynamic_pointer_cast** (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__gnu_cxx::__dynamic_pointer_cast** (const _FromType &__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__gnu_cxx::__reinterpret_pointer_cast** (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType **__gnu_cxx::__reinterpret_pointer_cast** (const _FromType &__arg)
- template<typename _ToType, typename _FromType>
  _ToType __gnu_cxx::__static_pointer_cast (_FromType *__arg)
- template<typename _ToType, typename _FromType>
  _ToType __gnu_cxx::__static_pointer_cast (const _FromType &__arg)

7.301.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/pointer.h>.

7.302 codecvt_specializations.h File Reference

Classes

- class std::codecvt< _InternT, _ExternT, encoding_state >
- struct __gnu_cxx::encoding_char_traits< _CharT >
- class __gnu_cxx::encoding_state
- class std::encoding_state

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- template<typename _Tp>
  size_t **std::__iconv_adaptor** (size_t(*__func)(iconv_t, _Tp, size_t *, char **, size_t *), iconv_t __cd, char **↩
  __inbuf, size_t *__inbytes, char **__outbuf, size_t *__outbytes)

Variables

- template<typename _InternT, typename _ExternT>
  locale::id **std::codecvt< _InternT, _ExternT, encoding_state >::id**

7.302.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.303 concurrence.h File Reference

Classes

- class __gnu_cxx::__scoped_lock

Namespaces

- namespace __gnu_cxx

Enumerations

- enum **_Lock_policy** { **_S_single** , **_S_mutex** , **_S_atomic** }

Functions

- void **__gnu_cxx::__throw_concurrence_lock_error** ()
- void **__gnu_cxx::__throw_concurrence_unlock_error** ()

Variables

- const _Lock_policy **__gnu_cxx::__default_lock_policy**

7.303.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.304 debug_allocator.h File Reference

Classes

- class __gnu_cxx::debug_allocator< _Alloc >

Namespaces

- namespace __gnu_cxx

7.304.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.305 enc_filebuf.h File Reference

Classes

- class __gnu_cxx::enc_filebuf< _CharT >

Namespaces

- namespace __gnu_cxx

7.305.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.306 extptr_allocator.h File Reference

Classes

- class __gnu_cxx::_ExtPtr_allocator< _Tp >

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _Tp>
  void **__gnu_cxx::swap** (_ExtPtr_allocator< _Tp > &__larg, _ExtPtr_allocator< _Tp > &__rarg)

7.306.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Author

Bob Walters

An example allocator which uses an alternative pointer type from bits/pointer.h. Supports test cases which confirm container support for alternative pointers.

7.307 malloc_allocator.h File Reference

Classes

- class __gnu_cxx::malloc_allocator< _Tp >

Namespaces

- namespace __gnu_cxx

7.307.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.308 mt_allocator.h File Reference

Classes

- struct __gnu_cxx::__common_pool_policy< _PoolTp, _Thread >
- class __gnu_cxx::__mt_alloc< _Tp, _Poolp >
- class __gnu_cxx::__mt_alloc_base< _Tp >
- struct __gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >
- class __gnu_cxx::__pool< false >
- class __gnu_cxx::__pool< true >
- struct __gnu_cxx::__pool_base

Namespaces

- namespace __gnu_cxx

Macros

- #define **__thread_default**

Typedefs

- typedef void(∗ **__gnu_cxx::__destroy_handler**) (void ∗)

Functions

- template<typename _Tp, typename _Poolp>
  bool **__gnu_cxx::operator==** (const __mt_alloc< _Tp, _Poolp > &, const __mt_alloc< _Tp, _Poolp > &)

7.308.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.309 assoc_container.hpp File Reference

Classes

- class __gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >
- class __gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >
- class __gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_↩ Hash, _Alloc >
- class __gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_↩ Policy, Store_Hash, _Alloc >
- class __gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >
- class __gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >
- class __gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_BRANCH_BASE**
- #define **PB_DS_CC_HASH_BASE**
- #define **PB_DS_GP_HASH_BASE**
- #define **PB_DS_HASH_BASE**
- #define **PB_DS_LU_BASE**
- #define **PB_DS_TREE_BASE**
- #define **PB_DS_TREE_NODE_AND_IT_TRAITS**
- #define **PB_DS_TRIE_BASE**
- #define **PB_DS_TRIE_NODE_AND_IT_TRAITS**

7.309.1 Detailed Description

Contains associative containers.

7.310 bin_search_tree_.hpp File Reference

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_NODE_CONSISTENT**(_Node)
- #define **PB_DS_BIN_TREE_NAME**
- #define **PB_DS_BIN_TREE_TRAITS_BASE**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_STRUCT_ONLY_ASSERT_VALID**(X)

Variables

- template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc>
  bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::node_allocator **__gnu_pbds::detail::bin_search_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::s_node_allocator**

7.310.1 Detailed Description

Contains an implementation class for binary search tree.

7.311 bin_search_tree_/node_iterators.hpp File Reference

Classes

- class __gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >
- class __gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_TREE_CONST_NODE_ITERATOR_CLASS_C_DEC**
- #define **PB_DS_TREE_NODE_ITERATOR_CLASS_C_DEC**

7.311.1 Detailed Description

Contains an implementation class for bin_search_tree_.

7.312 ov_tree_map_/node_iterators.hpp File Reference

Classes

- class __gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >
- class __gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_OV_TREE_CONST_NODE_ITERATOR_C_DEC**
- #define **PB_DS_OV_TREE_NODE_ITERATOR_C_DEC**

7.312.1 Detailed Description

Contains an implementation class for ov_tree_.

7.313 point_iterators.hpp File Reference

Classes

- class __gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >
- class __gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_TREE_CONST_IT_C_DEC**
- #define **PB_DS_TREE_CONST_ODIR_IT_C_DEC**
- #define **PB_DS_TREE_IT_C_DEC**
- #define **PB_DS_TREE_ODIR_IT_C_DEC**

7.313.1 Detailed Description

Contains an implementation class for bin_search_tree_.

7.314 bin_search_tree_/r_erase_fn_imps.hpp File Reference

7.314.1 Detailed Description

Contains an implementation class for bin_search_tree_.

7.315 pat_trie_/r_erase_fn_imps.hpp File Reference

7.315.1 Detailed Description

Contains an implementation class for pat_trie.

7.316 bin_search_tree_/rotate_fn_imps.hpp File Reference

7.316.1 Detailed Description

Contains imps for rotating nodes.

7.317 pat_trie_/rotate_fn_imps.hpp File Reference

7.317.1 Detailed Description

Contains imps for rotating nodes.

7.318 bin_search_tree_/traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >
- struct __gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >

Namespaces

- namespace __gnu_pbds

7.318.1 Detailed Description

Contains an implementation for bin_search_tree_.

7.319 branch_policy/traits.hpp File Reference

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_DEBUG_VERIFY**(_Cond)

7.319.1 Detailed Description

Contains an implementation class for tree-like classes.

7.320 ov_tree_map_/traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >

Namespaces

- namespace __gnu_pbds

7.320.1 Detailed Description

Contains an implementation class for ov_tree_.

7.321 pat_trie_/traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >
- struct __gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >

Namespaces

- namespace __gnu_pbds

7.321.1 Detailed Description

Contains an implementation class for pat_trie_.

7.322 rb_tree_map_/traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >

Namespaces

- namespace __gnu_pbds

7.322.1 Detailed Description

Contains an implementation for rb_tree_.

7.323 splay_tree_/traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >
- struct __gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >

Namespaces

- namespace __gnu_pbds

7.323.1 Detailed Description

Contains an implementation for splay_tree_.

7.324 binary_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_VALID**(X)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_DEBUG_VERIFY**(_Cond)
- #define **PB_DS_ENTRY_CMP_DEC**
- #define **PB_DS_RESIZE_POLICY_DEC**

Variables

- template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
  binary_heap< Value_Type, Cmp_Fn, _Alloc >::entry_allocator **__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >::s_entry_allocator**
- template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
  binary_heap< Value_Type, Cmp_Fn, _Alloc >::no_throw_copies_t **__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >::s_no_throw_copies_ind**

- template<typename Value_Type, typename Cmp_Fn, typename _Alloc>
  binary_heap< Value_Type, Cmp_Fn, _Alloc >::value_allocator **__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >::s_value_allocator**

7.324.1 Detailed Description

Contains an implementation class for a binary heap.

7.325 binary_heap_/const_iterator.hpp File Reference

Classes

- class __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_BIN_HEAP_CIT_BASE**

7.325.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

7.326 left_child_next_sibling_heap_/const_iterator.hpp File Reference

Classes

- class __gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_BASIC_HEAP_CIT_BASE**
- #define **PB_DS_CLASS_C_DEC**

7.326.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

7.327 unordered_iterator/const_iterator.hpp File Reference

7.327.1 Detailed Description

Contains an iterator class used for const ranging over the elements of the table.
This file is intended to be included inside a class definition, with PB_DS_CLASS_C_DEC defined to the name of the enclosing class.

7.328 bin_search_tree_/constructors_destructor_fn_imps.hpp File Reference

7.328.1 Detailed Description

Contains an implementation class for bin_search_tree_.

7.329 binary_heap_/constructors_destructor_fn_imps.hpp File Reference

7.329.1 Detailed Description

Contains an implementation class for binary_heap_.

7.330 binomial_heap_/constructors_destructor_fn_imps.hpp File Reference

7.330.1 Detailed Description

Contains an implementation for binomial_heap_.

7.331 binomial_heap_base_/constructors_destructor_fn_imps.hpp File Reference

7.331.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

7.332 left_child_next_sibling_heap_/constructors_destructor_fn_imps.hpp File Reference

7.332.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

7.333 ov_tree_map_/constructors_destructor_fn_imps.hpp File Reference

7.333.1 Detailed Description

Contains an implementation class for ov_tree_.

7.334 pairing_heap_/constructors_destructor_fn_imps.hpp File Reference

7.334.1 Detailed Description

Contains an implementation class for a pairing heap.

7.335 pat_trie_/constructors_destructor_fn_imps.hpp File Reference

7.335.1 Detailed Description

Contains an implementation class for pat_trie.

7.336 rb_tree_map_/constructors_destructor_fn_imps.hpp File Reference

7.336.1 Detailed Description

Contains an implementation for rb_tree_.

7.337 rc_binomial_heap_/constructors_destructor_fn_imps.hpp File Reference

7.337.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

7.338 splay_tree_/constructors_destructor_fn_imps.hpp File Reference

7.338.1 Detailed Description

Contains an implementation class for splay_tree_.

7.339 thin_heap_/constructors_destructor_fn_imps.hpp File Reference

7.339.1 Detailed Description

Contains an implementation for thin_heap_.

7.340 bin_search_tree_/debug_fn_imps.hpp File Reference

7.340.1 Detailed Description

Contains an implementation class for bin_search_tree_.

7.341 binary_heap_/debug_fn_imps.hpp File Reference

7.341.1 Detailed Description

Contains an implementation class for a binary_heap.

7.342 binomial_heap_/debug_fn_imps.hpp File Reference

7.342.1 Detailed Description

Contains an implementation for binomial_heap_.

7.343 binomial_heap_base_/debug_fn_imps.hpp File Reference

7.343.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

7.344 cc_hash_table_map_/debug_fn_imps.hpp File Reference

7.344.1 Detailed Description

Contains implementations of cc_ht_map_'s debug-mode functions.

7.345 gp_hash_table_map_/debug_fn_imps.hpp File Reference

7.345.1 Detailed Description

Contains implementations of gp_ht_map_'s debug-mode functions.

7.346 left_child_next_sibling_heap_/debug_fn_imps.hpp File Reference

7.346.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

7.347 list_update_map_/debug_fn_imps.hpp File Reference

7.347.1 Detailed Description

Contains implementations of cc_ht_map_'s debug-mode functions.

7.348 ov_tree_map_/debug_fn_imps.hpp File Reference

7.348.1 Detailed Description

Contains an implementation class for ov_tree_.

7.349 pairing_heap_/debug_fn_imps.hpp File Reference

7.349.1 Detailed Description

Contains an implementation class for a pairing heap.

7.350 pat_trie_/debug_fn_imps.hpp File Reference

7.350.1 Detailed Description

Contains an implementation class for pat_trie_.

7.351 rb_tree_map_/debug_fn_imps.hpp File Reference

7.351.1 Detailed Description

Contains an implementation for rb_tree_.

7.352 rc_binomial_heap_/debug_fn_imps.hpp File Reference

7.352.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

7.353 splay_tree_/debug_fn_imps.hpp File Reference

7.353.1 Detailed Description

Contains an implementation class for splay_tree_.

7.354 thin_heap_/debug_fn_imps.hpp File Reference

7.354.1 Detailed Description

Contains an implementation for thin_heap_.

7.355 entry_cmp.hpp File Reference

Classes

- struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >
- struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >
- struct __gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type

Namespaces

- namespace __gnu_pbds

7.355.1 Detailed Description

Contains an implementation class for a binary_heap.

7.356 entry_pred.hpp File Reference

Classes

- struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >
- struct __gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >

Namespaces

- namespace __gnu_pbds

7.356.1 Detailed Description

Contains an implementation class for a binary_heap.

7.357 bin_search_tree_/erase_fn_imps.hpp File Reference

7.357.1 Detailed Description

Contains an implementation class for bin_search_tree_.

7.358 binary_heap_/erase_fn_imps.hpp File Reference

7.358.1 Detailed Description

Contains an implementation class for a binary_heap.

7.359 binomial_heap_base_/erase_fn_imps.hpp File Reference

7.359.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

7.360 cc_hash_table_map_/erase_fn_imps.hpp File Reference

7.360.1 Detailed Description

Contains implementations of cc_ht_map_'s erase related functions.

7.361 gp_hash_table_map_/erase_fn_imps.hpp File Reference

7.361.1 Detailed Description

Contains implementations of gp_ht_map_'s erase related functions.

7.362 left_child_next_sibling_heap_/erase_fn_imps.hpp File Reference

7.362.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

7.363 list_update_map_/erase_fn_imps.hpp File Reference

7.363.1 Detailed Description

Contains implementations of lu_map_.

7.364 ov_tree_map_/erase_fn_imps.hpp File Reference

7.364.1 Detailed Description

Contains an implementation class for ov_tree_.

7.365 pairing_heap_/erase_fn_imps.hpp File Reference

7.365.1 Detailed Description

Contains an implementation class for a pairing heap.

7.366 pat_trie_/erase_fn_imps.hpp File Reference

7.366.1 Detailed Description

Contains an implementation class for pat_trie.

7.367 rb_tree_map_/erase_fn_imps.hpp File Reference

7.367.1 Detailed Description

Contains an implementation for rb_tree_.

7.368 rc_binomial_heap_/erase_fn_imps.hpp File Reference

7.368.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

7.369 splay_tree_/erase_fn_imps.hpp File Reference

7.369.1 Detailed Description

Contains an implementation class for splay_tree_.

7.370 thin_heap_/erase_fn_imps.hpp File Reference

7.370.1 Detailed Description

Contains an implementation for thin_heap_.

7.371 bin_search_tree_/find_fn_imps.hpp File Reference

7.371.1 Detailed Description

Contains an implementation class for bin_search_tree_.

7.372 binary_heap_/find_fn_imps.hpp File Reference

7.372.1 Detailed Description

Contains an implementation class for a binary_heap.

7.373 binomial_heap_base_/find_fn_imps.hpp File Reference

7.373.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

7.374 cc_hash_table_map_/find_fn_imps.hpp File Reference

7.374.1 Detailed Description

Contains implementations of cc_ht_map_'s find related functions.

7.375 gp_hash_table_map_/find_fn_imps.hpp File Reference

7.375.1 Detailed Description

Contains implementations of gp_ht_map_'s find related functions.

7.376 list_update_map_/find_fn_imps.hpp File Reference

7.376.1 Detailed Description

Contains implementations of lu_map_.

7.377 pairing_heap_/find_fn_imps.hpp File Reference

7.377.1 Detailed Description

Contains an implementation class for a pairing heap.

7.378 pat_trie_/find_fn_imps.hpp File Reference

7.378.1 Detailed Description

Contains an implementation class for pat_trie.

7.379 rb_tree_map_/find_fn_imps.hpp File Reference

7.379.1 Detailed Description

Contains an implementation for rb_tree_.

7.380 splay_tree_/find_fn_imps.hpp File Reference

7.380.1 Detailed Description

Contains an implementation class for splay_tree_.

7.381 thin_heap_/find_fn_imps.hpp File Reference

7.381.1 Detailed Description

Contains an implementation for thin_heap_.

7.382 bin_search_tree_/info_fn_imps.hpp File Reference

7.382.1 Detailed Description

Contains an implementation class for bin_search_tree_.

7.383 binary_heap_/info_fn_imps.hpp File Reference

7.383.1 Detailed Description

Contains an implementation class for a binary_heap.

7.384 cc_hash_table_map_/info_fn_imps.hpp File Reference

7.384.1 Detailed Description

Contains implementations of cc_ht_map_'s entire container info related functions.

7.385 gp_hash_table_map_/info_fn_imps.hpp File Reference

7.385.1 Detailed Description

Contains implementations of gp_ht_map_'s entire container info related functions.

7.386 left_child_next_sibling_heap_/info_fn_imps.hpp File Reference

7.386.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

7.387 list_update_map_/info_fn_imps.hpp File Reference

7.387.1 Detailed Description

Contains implementations of lu_map_.

7.388 ov_tree_map_/info_fn_imps.hpp File Reference

7.388.1 Detailed Description

Contains an implementation class for ov_tree_.

7.389 pat_trie_/info_fn_imps.hpp File Reference

7.389.1 Detailed Description

Contains an implementation class for pat_trie.

7.390 rb_tree_map_/info_fn_imps.hpp File Reference

7.390.1 Detailed Description

Contains an implementation for rb_tree_.

7.391 splay_tree_/info_fn_imps.hpp File Reference

7.391.1 Detailed Description

Contains an implementation.

7.392 bin_search_tree_/insert_fn_imps.hpp File Reference

7.392.1 Detailed Description

Contains an implementation class for bin_search_tree_.

7.393 binary_heap_/insert_fn_imps.hpp File Reference

7.393.1 Detailed Description

Contains an implementation class for a binary_heap.

7.394 binomial_heap_base_/insert_fn_imps.hpp File Reference

7.394.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

7.395 cc_hash_table_map_/insert_fn_imps.hpp File Reference

7.395.1 Detailed Description

Contains implementations of cc_ht_map_'s insert related functions.

7.396 gp_hash_table_map_/insert_fn_imps.hpp File Reference

7.396.1 Detailed Description

Contains implementations of gp_ht_map_'s insert related functions.

7.397 left_child_next_sibling_heap_/insert_fn_imps.hpp File Reference

7.397.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

7.398 list_update_map_/insert_fn_imps.hpp File Reference

7.398.1 Detailed Description

Contains implementations of lu_map_.

7.399 ov_tree_map_/insert_fn_imps.hpp File Reference

7.399.1 Detailed Description

Contains an implementation class for ov_tree_.

7.400 pairing_heap_/insert_fn_imps.hpp File Reference

7.400.1 Detailed Description

Contains an implementation class for a pairing heap.

7.401 rb_tree_map_/insert_fn_imps.hpp File Reference

7.401.1 Detailed Description

Contains an implementation for rb_tree_.

7.402 rc_binomial_heap_/insert_fn_imps.hpp File Reference

7.402.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

7.403 splay_tree_/insert_fn_imps.hpp File Reference

7.403.1 Detailed Description

Contains an implementation class for splay_tree_.

7.404 thin_heap_/insert_fn_imps.hpp File Reference

7.404.1 Detailed Description

Contains an implementation for thin_heap_.

7.405 bin_search_tree_/iterators_fn_imps.hpp File Reference

7.405.1 Detailed Description

Contains an implementation class for bin_search_tree_.

7.406 binary_heap_/iterators_fn_imps.hpp File Reference

7.406.1 Detailed Description

Contains an implementation class for a binary_heap.

7.407 cc_hash_table_map_/iterators_fn_imps.hpp File Reference

7.407.1 Detailed Description

Contains implementations of cc_ht_map_'s iterators related functions, e.g., begin().

7.408 left_child_next_sibling_heap_/iterators_fn_imps.hpp File Reference

7.408.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

7.409 list_update_map_/iterators_fn_imps.hpp File Reference

7.409.1 Detailed Description

Contains implementations of lu_map_.

7.410 ov_tree_map_/iterators_fn_imps.hpp File Reference

7.410.1 Detailed Description

Contains an implementation class for ov_tree_.

7.411 pat_trie_/iterators_fn_imps.hpp File Reference

7.411.1 Detailed Description

Contains an implementation class for pat_trie.

7.412 binary_heap_/point_const_iterator.hpp File Reference

Classes

- class __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >

Namespaces

- namespace __gnu_pbds

7.412.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

7.413 left_child_next_sibling_heap_/point_const_iterator.hpp File Reference

Classes

- class __gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

7.413.1 Detailed Description

Contains an iterator class returned by the table's const find and insert methods.

7.414 unordered_iterator/point_const_iterator.hpp File Reference

7.414.1 Detailed Description

Contains an iterator class returned by the tables' const find and insert methods.

- This file is intended to be included inside a class definition, with PB_DS_CLASS_C_DEC defined to the name of the enclosing class.

7.415 bin_search_tree_/policy_access_fn_imps.hpp File Reference

7.415.1 Detailed Description

Contains an implementation class for bin_search_tree_.

7.416 binary_heap_/policy_access_fn_imps.hpp File Reference

7.416.1 Detailed Description

Contains an implementation class for a binary_heap.

7.417 cc_hash_table_map_/policy_access_fn_imps.hpp File Reference

7.417.1 Detailed Description

Contains implementations of cc_ht_map_'s policy access functions.

7.418 gp_hash_table_map_/policy_access_fn_imps.hpp File Reference

7.418.1 Detailed Description

Contains implementations of gp_ht_map_'s policy agpess functions.

7.419 left_child_next_sibling_heap_/policy_access_fn_imps.hpp File Reference

7.419.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

7.420 ov_tree_map_/policy_access_fn_imps.hpp File Reference

7.420.1 Detailed Description

Contains an implementation class for ov_tree.

7.421 pat_trie_/policy_access_fn_imps.hpp File Reference

7.421.1 Detailed Description

Contains an implementation class for pat_trie.

7.422 resize_policy.hpp File Reference

Classes

- class __gnu_pbds::detail::resize_policy< _Tp >

Namespaces

- namespace __gnu_pbds

Variables

- template<typename _Tp>
  const _Tp **__gnu_pbds::detail::resize_policy< _Tp >::min_size**

7.422.1 Detailed Description

Contains an implementation class for a binary_heap.

7.423 bin_search_tree_/split_join_fn_imps.hpp File Reference

7.423.1 Detailed Description

Contains an implementation class for bin_search_tree_.

7.424 binary_heap_/split_join_fn_imps.hpp File Reference

7.424.1 Detailed Description

Contains an implementation class for a binary_heap.

7.425 binomial_heap_base_/split_join_fn_imps.hpp File Reference

7.425.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

7.426 ov_tree_map_/split_join_fn_imps.hpp File Reference

7.426.1 Detailed Description

Contains an implementation class for ov_tree_.

7.427 pairing_heap_/split_join_fn_imps.hpp File Reference

7.427.1 Detailed Description

Contains an implementation class for a pairing heap.

7.428 rb_tree_map_/split_join_fn_imps.hpp File Reference

7.428.1 Detailed Description

Contains an implementation for rb_tree_.

7.429 rc_binomial_heap_/split_join_fn_imps.hpp File Reference

7.429.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

7.430 splay_tree_/split_join_fn_imps.hpp File Reference

7.430.1 Detailed Description

Contains an implementation class for splay_tree_.

7.431 thin_heap_/split_join_fn_imps.hpp File Reference

7.431.1 Detailed Description

Contains an implementation for thin_heap_.

7.432 binary_heap_/trace_fn_imps.hpp File Reference

7.432.1 Detailed Description

Contains an implementation class for a binary_heap.

7.433 cc_hash_table_map_/trace_fn_imps.hpp File Reference

7.433.1 Detailed Description

Contains implementations of cc_ht_map_'s trace-mode functions.

7.434 gp_hash_table_map_/trace_fn_imps.hpp File Reference

7.434.1 Detailed Description

Contains implementations of gp_ht_map_'s trace-mode functions.

7.435 left_child_next_sibling_heap_/trace_fn_imps.hpp File Reference

7.435.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

7.436 list_update_map_/trace_fn_imps.hpp File Reference

7.436.1 Detailed Description

Contains implementations of lu_map_.

7.437 pat_trie_/trace_fn_imps.hpp File Reference

7.437.1 Detailed Description

Contains an implementation class for pat_trie_.

7.438 rc_binomial_heap_/trace_fn_imps.hpp File Reference

7.438.1 Detailed Description

Contains an implementation for rc_binomial_heap_.

7.439 thin_heap_/trace_fn_imps.hpp File Reference

7.439.1 Detailed Description

Contains an implementation class for left_child_next_sibling_heap_.

7.440 binomial_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

7.440.1 Detailed Description

Contains an implementation class for a binomial heap.

7.441 binomial_heap_base_.hpp File Reference

Classes

- class __gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_BASE_NODE_CONSISTENT**(_Node, _Bool)
- #define **PB_DS_ASSERT_VALID_COND**(X, _StrictlyBinomial)
- #define **PB_DS_B_HEAP_BASE**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

7.441.1 Detailed Description

Contains an implementation class for a base of binomial heaps.

7.442 branch_policy.hpp File Reference

Classes

- struct __gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc >
- struct __gnu_pbds::detail::branch_policy< Node_CItr, Node_CItr, _Alloc >

Namespaces

- namespace __gnu_pbds

7.442.1 Detailed Description

Contains a base class for branch policies.

7.443 null_node_metadata.hpp File Reference

Classes

- struct __gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >

Namespaces

- namespace __gnu_pbds

7.443.1 Detailed Description

Contains an implementation class for tree-like classes.

7.444 cc_ht_map_.hpp File Reference

Classes

- class __gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CC_HASH_NAME**
- #define **PB_DS_CC_HASH_TRAITS_BASE**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_GEN_POS**
- #define **PB_DS_HASH_EQ_FN_C_DEC**
- #define **PB_DS_RANGED_HASH_FN_C_DEC**

Variables

- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Hash_Fn, typename Resize_Policy>
  cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::const↩ _iterator **__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩ Hash_Fn, Resize_Policy >::s_const_end_it**
- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Hash_Fn, typename Resize_Policy>
  cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::iterator **__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::s_end_it**
- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Hash_Fn, typename Resize_Policy>
  cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::entry_↩ allocator **__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_↩ Hash_Fn, Resize_Policy >::s_entry_allocator**
- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Hash_Fn, typename Resize_Policy>
  cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::entry↩ _pointer_allocator **__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >::s_entry_pointer_allocator**

7.444.1 Detailed Description

Contains an implementation class for cc_ht_map_.

7.445 cmp_fn_imps.hpp File Reference

7.445.1 Detailed Description

Contains implementations of cc_ht_map_'s entire container comparison related functions.

7.446 cond_key_dtor_entry_dealtor.hpp File Reference

Classes

- class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >

Namespaces

- namespace __gnu_pbds

7.446.1 Detailed Description

Contains a conditional key destructor, used for exception handling.

7.447 cc_hash_table_map_/constructor_destructor_fn_imps.hpp File Reference

7.447.1 Detailed Description

Contains implementations of cc_ht_map_'s constructors, destructor, and related functions.

7.448 gp_hash_table_map_/constructor_destructor_fn_imps.hpp File Reference

7.448.1 Detailed Description

Contains implementations of gp_ht_map_'s constructors, destructor, and related functions.

7.449 list_update_map_/constructor_destructor_fn_imps.hpp File Reference

7.450 cc_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp File Reference

7.450.1 Detailed Description

Contains implementations of cc_ht_map_'s constructors, destructor, and related functions.

7.451 gp_hash_table_map_/constructor_destructor_no_store_hash_fn_imps.hpp File Reference

7.451.1 Detailed Description

Contains implementations of gp_ht_map_'s constructors, destructor, and related functions.

7.452 cc_hash_table_map_/constructor_destructor_store_hash_fn_imps.hpp File Reference

7.452.1 Detailed Description

Contains implementations of cc_ht_map_'s constructors, destructor, and related functions.

7.453 gp_hash_table_map_/constructor_destructor_store_hash_fn_imps.hpp File Reference

7.453.1 Detailed Description

Contains implementations of gp_ht_map_'s constructors, destructor, and related functions.

7.454 cc_hash_table_map_/debug_no_store_hash_fn_imps.hpp File Reference

7.454.1 Detailed Description

Contains implementations of cc_ht_map_'s debug-mode functions.

7.455 gp_hash_table_map_/debug_no_store_hash_fn_imps.hpp File Reference

7.455.1 Detailed Description

Contains implementations of gp_ht_map_'s debug-mode functions.

7.456 cc_hash_table_map_/debug_store_hash_fn_imps.hpp File Reference

7.456.1 Detailed Description

Contains implementations of cc_ht_map_'s debug-mode functions.

7.457 gp_hash_table_map_/debug_store_hash_fn_imps.hpp File Reference

7.457.1 Detailed Description

Contains implementations of gp_ht_map_'s debug-mode functions.

7.458 entry_list_fn_imps.hpp File Reference

7.458.1 Detailed Description

Contains implementations of cc_ht_map_'s entry-list related functions.

7.459 cc_hash_table_map_/erase_no_store_hash_fn_imps.hpp File Reference

7.459.1 Detailed Description

Contains implementations of cc_ht_map_'s erase related functions, when the hash value is not stored.

7.460 gp_hash_table_map_/erase_no_store_hash_fn_imps.hpp File Reference

7.460.1 Detailed Description

Contains implementations of gp_ht_map_'s erase related functions, when the hash value is not stored.

7.461 cc_hash_table_map_/erase_store_hash_fn_imps.hpp File Reference

7.461.1 Detailed Description

Contains implementations of cc_ht_map_'s erase related functions, when the hash value is stored.

7.462 gp_hash_table_map_/erase_store_hash_fn_imps.hpp File Reference

7.462.1 Detailed Description

Contains implementations of gp_ht_map_'s erase related functions, when the hash value is stored.

7.463 cc_hash_table_map_/find_store_hash_fn_imps.hpp File Reference

7.463.1 Detailed Description

Contains implementations of cc_ht_map_'s find related functions, when the hash value is stored.

7.464 gp_hash_table_map_/find_store_hash_fn_imps.hpp File Reference

7.464.1 Detailed Description

Contains implementations of gp_ht_map_'s insert related functions, when the hash value is stored.

7.465 cc_hash_table_map_/insert_no_store_hash_fn_imps.hpp File Reference

7.465.1 Detailed Description

Contains implementations of cc_ht_map_'s insert related functions, when the hash value is not stored.

7.466 gp_hash_table_map_/insert_no_store_hash_fn_imps.hpp File Reference

7.466.1 Detailed Description

Contains implementations of gp_ht_map_'s insert related functions, when the hash value is not stored.

7.467 cc_hash_table_map_/insert_store_hash_fn_imps.hpp File Reference

7.467.1 Detailed Description

Contains implementations of cc_ht_map_'s insert related functions, when the hash value is stored.

7.468 gp_hash_table_map_/insert_store_hash_fn_imps.hpp File Reference

7.468.1 Detailed Description

Contains implementations of gp_ht_map_'s find related functions, when the hash value is stored.

7.469 cc_hash_table_map_/resize_fn_imps.hpp File Reference

7.469.1 Detailed Description

Contains implementations of cc_ht_map_'s resize related functions.

7.470 gp_hash_table_map_/resize_fn_imps.hpp File Reference

7.470.1 Detailed Description

Contains implementations of gp_ht_map_'s resize related functions.

7.471 cc_hash_table_map_/resize_no_store_hash_fn_imps.hpp File Reference

7.471.1 Detailed Description

Contains implementations of cc_ht_map_'s resize related functions, when the hash value is not stored.

7.472 gp_hash_table_map_/resize_no_store_hash_fn_imps.hpp File Reference

7.472.1 Detailed Description

Contains implementations of gp_ht_map_'s resize related functions, when the hash value is not stored.

7.473 cc_hash_table_map_/resize_store_hash_fn_imps.hpp File Reference

7.473.1 Detailed Description

Contains implementations of cc_ht_map_'s resize related functions, when the hash value is stored.

7.474 gp_hash_table_map_/resize_store_hash_fn_imps.hpp File Reference

7.474.1 Detailed Description

Contains implementations of gp_ht_map_'s resize related functions, when the hash value is stored.

7.475 size_fn_imps.hpp File Reference

7.475.1 Detailed Description

Contains implementations of cc_ht_map_'s entire container size related functions.

7.476 cond_dealtor.hpp File Reference

Classes

- class __gnu_pbds::detail::cond_dealtor< Entry, _Alloc >

Namespaces

- namespace __gnu_pbds

Variables

- template<typename Entry, class _Alloc>
  cond_dealtor< Entry, _Alloc >::entry_allocator **__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >::s_alloc**

7.476.1 Detailed Description

Contains a conditional deallocator.

7.477 container_base_dispatch.hpp File Reference

Classes

- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >
- struct __gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_VALID**(X)
- #define **PB_DS_CHECK_KEY_DOES_NOT_EXIST**(_Key)
- #define **PB_DS_CHECK_KEY_EXISTS**(_Key)
- #define **PB_DS_DATA_FALSE_INDICATOR**
- #define **PB_DS_DATA_TRUE_INDICATOR**
- #define **PB_DS_DEBUG_VERIFY**(_Cond)
- #define **PB_DS_EP2VP**(X)
- #define **PB_DS_EP2VP**(X)
- #define **PB_DS_V2F**(X)
- #define **PB_DS_V2F**(X)
- #define **PB_DS_V2S**(X)
- #define **PB_DS_V2S**(X)

7.477.1 Detailed Description

Contains associative container dispatching.

7.478 debug_map_base.hpp File Reference

7.478.1 Detailed Description

Contains a debug-mode base for all maps.

7.479 eq_by_less.hpp File Reference

Classes

- struct __gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >

Namespaces

- namespace __gnu_pbds

7.479.1 Detailed Description

Contains an equivalence function.

7.480 hash_eq_fn.hpp File Reference

Classes

- struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >
- struct __gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >

Namespaces

- namespace __gnu_pbds

7.480.1 Detailed Description

Contains 2 equivalence functions, one employing a hash value, and one ignoring it.

7.481 find_no_store_hash_fn_imps.hpp File Reference

7.481.1 Detailed Description

Contains implementations of gp_ht_map_'s find related functions, when the hash value is not stored.

7.482 gp_ht_map_.hpp File Reference

Classes

- class __gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_GEN_POS**
- #define **PB_DS_GP_HASH_NAME**
- #define **PB_DS_GP_HASH_TRAITS_BASE**
- #define **PB_DS_HASH_EQ_FN_C_DEC**
- #define **PB_DS_RANGED_PROBE_FN_C_DEC**

Variables

- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Probe_Fn, typename Probe_Fn, typename Resize_Policy>
  gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::const_iterator **__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::s_const_end_it**
- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Probe_Fn, typename Probe_Fn, typename Resize_Policy>
  gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::iterator **__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb↩ _Probe_Fn, Probe_Fn, Resize_Policy >::s_end_it**
- template<typename Key, typename Mapped, typename Hash_Fn, typename Eq_Fn, typename _Alloc, bool Store_Hash, typename Comb↩ _Probe_Fn, typename Probe_Fn, typename Resize_Policy>
  gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::entry_allocator **__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >::s_entry_allocator**

7.482.1 Detailed Description

Contains an implementation class for general probing hash.

7.483 iterator_fn_imps.hpp File Reference

7.483.1 Detailed Description

Contains implementations of gp_ht_map_'s iterators related functions, e.g., begin().

7.484 direct_mask_range_hashing_imp.hpp File Reference

7.484.1 Detailed Description

Contains a range-hashing policy implementation

7.485 direct_mod_range_hashing_imp.hpp File Reference

7.485.1 Detailed Description

Contains a range-hashing policy implementation

7.486 linear_probe_fn_imp.hpp File Reference

7.486.1 Detailed Description

Contains a probe policy implementation

7.487 mask_based_range_hashing.hpp File Reference

Classes

- class __gnu_pbds::detail::mask_based_range_hashing< Size_Type >

Namespaces

- namespace __gnu_pbds

Variables

- template<typename Size_Type>
  const mask_based_range_hashing< Size_Type >::size_type **__gnu_pbds::detail::mask_based_range_hashing< Size_Type >::s_highest_bit_1**
- template<typename Size_Type>
  const mask_based_range_hashing< Size_Type >::size_type **__gnu_pbds::detail::mask_based_range_hashing< Size_Type >::s_num_bits_in_size_type**

7.487.1 Detailed Description

Contains a range hashing policy base.

7.488 mod_based_range_hashing.hpp File Reference

Classes

- class __gnu_pbds::detail::mod_based_range_hashing< Size_Type >

Namespaces

- namespace __gnu_pbds

7.488.1 Detailed Description

Contains a range hashing policy base.

7.489 probe_fn_base.hpp File Reference

Classes

- class __gnu_pbds::detail::probe_fn_base< _Alloc >

Namespaces

- namespace __gnu_pbds

7.489.1 Detailed Description

Contains a probe policy base.

7.490 quadratic_probe_fn_imp.hpp File Reference

7.490.1 Detailed Description

Contains a probe policy implementation

7.491 ranged_hash_fn.hpp File Reference

Classes

- class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >
- class __gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >
- class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >
- class __gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**

7.491.1 Detailed Description

Contains a unified ranged hash functor, allowing the hash tables to deal with a single class for ranged hashing.

7.492 ranged_probe_fn.hpp File Reference

Classes

- class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >
- class __gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >
- class __gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**

7.492.1 Detailed Description

Contains a unified ranged probe functor, allowing the probe tables to deal with a single class for ranged probeing.

7.493 sample_probe_fn.hpp File Reference

Classes

- class __gnu_pbds::sample_probe_fn

Namespaces

- namespace __gnu_pbds

7.493.1 Detailed Description

Contains a sample probe policy.

7.494 sample_range_hashing.hpp File Reference

Classes

- class __gnu_pbds::sample_range_hashing

Namespaces

- namespace __gnu_pbds

7.494.1 Detailed Description

Contains a range hashing policy.

7.495 sample_ranged_hash_fn.hpp File Reference

Classes

- class __gnu_pbds::sample_ranged_hash_fn

Namespaces

- namespace __gnu_pbds

7.495.1 Detailed Description

Contains a ranged hash policy.

7.496 sample_ranged_probe_fn.hpp File Reference

Classes

- class __gnu_pbds::sample_ranged_probe_fn

Namespaces

- namespace __gnu_pbds

7.496.1 Detailed Description

Contains a ranged probe policy.

7.497 left_child_next_sibling_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

Variables

- template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>
  left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::no_throw_copies_t **__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::s_no_throw_copies_ind**
- template<typename Value_Type, typename Cmp_Fn, typename Node_Metadata, typename _Alloc>
  left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::node_allocator **__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >::s_node_allocator**

7.497.1 Detailed Description

Contains an implementation class for a basic heap.

7.498 left_child_next_sibling_heap_/node.hpp File Reference

Classes

- struct __gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc >

Namespaces

- namespace __gnu_pbds

7.498.1 Detailed Description

Contains an implementation struct for this type of heap's node.

7.499 rb_tree_map_/node.hpp File Reference

Classes

- struct __gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >

Namespaces

- namespace __gnu_pbds

7.499.1 Detailed Description

Contains an implementation for rb_tree_.

7.500 splay_tree_/node.hpp File Reference

Classes

- struct __gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >

Namespaces

- namespace __gnu_pbds

7.500.1 Detailed Description

Contains an implementation struct for splay_tree_'s node.

7.501 entry_metadata_base.hpp File Reference

Namespaces

- namespace __gnu_pbds

7.501.1 Detailed Description

Contains an implementation for a list update map.

7.502 lu_map_.hpp File Reference

Classes

- class __gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_GEN_POS**
- #define **PB_DS_LU_NAME**
- #define **PB_DS_LU_TRAITS_BASE**

Variables

- template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
  lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::entry_allocator **__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::s_entry_allocator**
- template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
  Eq_Fn **__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::s_eq_fn**
- template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
  type_to_type< typename lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::update_metadata > __↩
  gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::s_metadata_type_indicator
- template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
  null_type **__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::s_null_type**
- template<typename Key, typename Mapped, typename Eq_Fn, typename _Alloc, typename Update_Policy>
  Update_Policy **__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >::s_update_↩
  policy**

7.502.1 Detailed Description

Contains a list update map.

7.503 lu_counter_metadata.hpp File Reference

Classes

- class __gnu_pbds::detail::lu_counter_metadata< Size_Type >
- class __gnu_pbds::detail::lu_counter_policy_base< Size_Type >

Namespaces

- namespace __gnu_pbds

7.503.1 Detailed Description

Contains implementation of a lu counter policy's metadata.

7.504 sample_update_policy.hpp File Reference

Classes

- struct __gnu_pbds::sample_update_policy

Namespaces

- namespace __gnu_pbds

7.504.1 Detailed Description

Contains a sample policy for list update containers.

7.505 ov_tree_map_.hpp File Reference

Classes

- class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >
- class __gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CONST_NODE_ITERATOR_NAME**
- #define **PB_DS_OV_TREE_NAME**
- #define **PB_DS_OV_TREE_TRAITS_BASE**

Variables

- template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc> ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::metadata_allocator **__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::s_metadata_alloc**
- template<typename Key, typename Mapped, typename Cmp_Fn, typename Node_And_It_Traits, typename _Alloc> ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::value_allocator **__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::s_value_alloc**

7.505.1 Detailed Description

Contains an implementation class for ov_tree.

7.506 pairing_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_NODE_CONSISTENT**(_Node, _Bool)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_P_HEAP_BASE**

7.506.1 Detailed Description

Contains an implementation class for a pairing heap.

7.507 insert_join_fn_imps.hpp File Reference

7.507.1 Detailed Description

Contains an implementation class for pat_trie.

7.508 pat_trie_.hpp File Reference

Classes

- class __gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_NODE_VALID**(X)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_PAT_TRIE_NAME**
- #define **PB_DS_PAT_TRIE_TRAITS_BASE**
- #define **PB_DS_RECURSIVE_COUNT_LEAFS**(X)

Variables

- template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
  pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::head_allocator **__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::s_head_allocator**
- template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
  pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::inode_allocator **__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::s_inode_allocator**
- template<typename Key, typename Mapped, typename Node_And_It_Traits, typename _Alloc>
  pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::leaf_allocator **__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >::s_leaf_allocator**

7.508.1 Detailed Description

Contains an implementation class for a patricia tree.

7.509 pat_trie_base.hpp File Reference

Classes

- class __gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
- struct __gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >
- struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >
- class __gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >
- struct __gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >
- struct __gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >
- struct __gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >
- struct __gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >
- class __gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >
- class __gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >
- struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator
- struct __gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator
- struct __gnu_pbds::detail::pat_trie_base

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CONST_IT_C_DEC**
- #define **PB_DS_CONST_ODIR_IT_C_DEC**
- #define **PB_DS_IT_C_DEC**
- #define **PB_DS_ODIR_IT_C_DEC**
- #define **PB_DS_PAT_TRIE_NODE_CONST_ITERATOR_C_DEC**
- #define **PB_DS_PAT_TRIE_NODE_ITERATOR_C_DEC**

7.509.1 Detailed Description

Contains the base class for a patricia tree.

7.510 split_fn_imps.hpp File Reference

7.510.1 Detailed Description

Contains an implementation class for pat_trie.

7.511 synth_access_traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_SYNTH_E_ACCESS_TRAITS_C_DEC**
- #define **PB_DS_SYNTH_E_ACCESS_TRAITS_T_DEC**

Variables

- template<typename Type_Traits, bool Set, typename _ATraits>
  integral_constant< int, Set > **__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >::s_set_ind**

7.511.1 Detailed Description

Contains an implementation class for a patricia tree.

7.512 update_fn_imps.hpp File Reference

7.512.1 Detailed Description

Contains an implementation class for pat_trie_.

7.513 priority_queue_base_dispatch.hpp File Reference

Classes

- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >
- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >
- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >
- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >
- struct __gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_VALID**(X)
- #define **PB_DS_DEBUG_VERIFY**(_Cond)

7.513.1 Detailed Description

Contains an pqiative container dispatching base.

7.514 rb_tree_.hpp File Reference

Classes

- class __gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_RB_TREE_BASE**
- #define **PB_DS_RB_TREE_BASE_NAME**
- #define **PB_DS_RB_TREE_NAME**
- #define **PB_DS_STRUCT_ONLY_ASSERT_VALID**(X)

7.514.1 Detailed Description

Contains an implementation for Red Black trees.

7.515 rc.hpp File Reference

Classes

- class __gnu_pbds::detail::rc< _Node, _Alloc >

Namespaces

- namespace __gnu_pbds

7.515.1 Detailed Description

Contains a redundant (binary counter).

7.516 rc_binomial_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_RC_C_DEC**

7.516.1 Detailed Description

Contains an implementation for redundant-counter binomial heap.

7.517 cc_hash_max_collision_check_resize_trigger_imp.hpp File Reference

7.517.1 Detailed Description

Contains a resize trigger implementation.

7.518 hash_exponential_size_policy_imp.hpp File Reference

7.518.1 Detailed Description

Contains a resize size policy implementation.

7.519 hash_load_check_resize_trigger_imp.hpp File Reference

7.519.1 Detailed Description

Contains a resize trigger implementation.

7.520 hash_load_check_resize_trigger_size_base.hpp File Reference

Classes

- class __gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >

Namespaces

- namespace __gnu_pbds

7.520.1 Detailed Description

Contains an base holding size for some resize policies.

7.521 hash_prime_size_policy_imp.hpp File Reference

7.521.1 Detailed Description

Contains a resize size policy implementation.

7.522 hash_standard_resize_policy_imp.hpp File Reference

7.522.1 Detailed Description

Contains a resize policy implementation.

7.523 sample_resize_policy.hpp File Reference

Classes

- class __gnu_pbds::sample_resize_policy

Namespaces

- namespace __gnu_pbds

7.523.1 Detailed Description

Contains a sample resize policy for hash tables.

7.524 sample_resize_trigger.hpp File Reference

Classes

- class __gnu_pbds::sample_resize_trigger

Namespaces

- namespace __gnu_pbds

7.524.1 Detailed Description

Contains a sample resize trigger policy class.

7.525 sample_size_policy.hpp File Reference

Classes

- class __gnu_pbds::sample_size_policy

Namespaces

- namespace __gnu_pbds

7.525.1 Detailed Description

Contains a sample size resize-policy.

7.526 splay_fn_imps.hpp File Reference

7.526.1 Detailed Description

Contains an implementation class for splay_tree_.

7.527 splay_tree_.hpp File Reference

Classes

- class __gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_BASE_NODE_CONSISTENT**(_Node)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_S_TREE_BASE**
- #define **PB_DS_S_TREE_BASE_NAME**
- #define **PB_DS_S_TREE_NAME**

7.527.1 Detailed Description

Contains an implementation class for splay trees.

7.528 standard_policies.hpp File Reference

Classes

- struct __gnu_pbds::detail::default_comb_hash_fn
- struct __gnu_pbds::detail::default_eq_fn< Key >
- struct __gnu_pbds::detail::default_hash_fn< Key >
- struct __gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >
- struct __gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >
- struct __gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >
- struct __gnu_pbds::detail::default_update_policy

Namespaces

- namespace __gnu_pbds

Macros

- #define **__dtrie_alloc**
- #define **__dtrie_string**

Enumerations

- enum { **default_store_hash** }

7.528.1 Detailed Description

Contains standard policies for containers.

7.528.2 Enumeration Type Documentation

anonymous enum

`anonymous enum`
Enumeration for default behavior of stored hash data.

7.529 thin_heap_.hpp File Reference

Classes

- class __gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_AUX_NULL**(X)
- #define **PB_DS_ASSERT_NODE_CONSISTENT**(_Node, _Bool)
- #define **PB_DS_BASE_T_P**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

Enumerations

- enum { **num_distinct_rank_bounds** }

Variables

- static const std::size_t **__gnu_pbds::detail::g_a_rank_bounds** [num_distinct_rank_bounds]

7.529.1 Detailed Description

Contains an implementation class for a thin heap.

7.530 tree_policy/node_metadata_selector.hpp File Reference

Classes

- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, false >
- struct __gnu_pbds::detail::tree_metadata_helper< Node_Update, true >
- struct __gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >

Namespaces

- namespace __gnu_pbds

7.530.1 Detailed Description

Contains an implementation class for trees.

7.531 trie_policy/node_metadata_selector.hpp File Reference

Classes

- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, false >
- struct __gnu_pbds::detail::trie_metadata_helper< Node_Update, true >
- struct __gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >

Namespaces

- namespace __gnu_pbds

7.531.1 Detailed Description

Contains an implementation class for tries.

7.532 tree_policy/order_statistics_imp.hpp File Reference

7.532.1 Detailed Description

Contains forward declarations for order_statistics_key

7.533 trie_policy/order_statistics_imp.hpp File Reference

7.533.1 Detailed Description

Contains forward declarations for order_statistics_key

7.534 sample_tree_node_update.hpp File Reference

Classes

- class __gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

7.534.1 Detailed Description

Contains a sample node update functor.

7.535 tree_trace_base.hpp File Reference

7.535.1 Detailed Description

Contains tree-related policies.

7.536 prefix_search_node_update_imp.hpp File Reference

7.536.1 Detailed Description

Contains an implementation of prefix_search_node_update.

7.537 sample_trie_access_traits.hpp File Reference

Classes

- struct __gnu_pbds::sample_trie_access_traits

Namespaces

- namespace __gnu_pbds

7.537.1 Detailed Description

Contains a sample probe policy.

7.538 sample_trie_node_update.hpp File Reference

Classes

- class __gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >

Namespaces

- namespace __gnu_pbds

7.538.1 Detailed Description

Contains a sample node update functor.

7.539 trie_policy_base.hpp File Reference

Classes

- class __gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

7.539.1 Detailed Description

Contains an implementation of trie_policy_base.

7.540 trie_string_access_traits_imp.hpp File Reference

7.540.1 Detailed Description

Contains a policy for extracting character positions from a string for a vector-based PATRICIA tree

7.541 type_utils.hpp File Reference

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_STATIC_ASSERT**(UNIQUE, E)

Typedefs

- typedef std::tr1::integral_constant< int, 0 > **__gnu_pbds::detail::false_type**
- typedef std::tr1::integral_constant< int, 1 > **__gnu_pbds::detail::true_type**

7.541.1 Detailed Description

Contains utilities for handling types. All of these classes are based on Modern C++ by Andrei Alxandrescu.

7.542 types_traits.hpp File Reference

Classes

- struct __gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >
- struct __gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >
- struct __gnu_pbds::detail::no_throw_copies< Key, Mapped >
- struct __gnu_pbds::detail::no_throw_copies< Key, null_type >
- struct __gnu_pbds::detail::rebind_traits< _Alloc, T >

- struct __gnu_pbds::detail::select_value_type< Key, Mapped >
- struct __gnu_pbds::detail::select_value_type< Key, null_type >
- struct __gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >
- struct __gnu_pbds::detail::stored_data< _Tv, _Th, false >
- struct __gnu_pbds::detail::stored_hash< _Th >
- struct __gnu_pbds::detail::stored_value< _Tv >
- struct __gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >

Namespaces

- namespace __gnu_pbds

Variables

- template<typename Key, typename _Alloc, bool Store_Hash>
  null_type **__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >::s_null_type**

7.542.1 Detailed Description

Contains a traits class of types used by containers.

7.543 iterator.hpp File Reference

7.543.1 Detailed Description

Contains an iterator_ class used for ranging over the elements of the table.
This file is intended to be included inside a class definition, with PB_DS_CLASS_C_DEC defined to the name of the enclosing class.

7.544 point_iterator.hpp File Reference

7.544.1 Detailed Description

Contains an iterator class returned by the tables' find and insert methods.
This file is intended to be included inside a class definition, with PB_DS_CLASS_C_DEC defined to the name of the enclosing class.

7.545 exception.hpp File Reference

Classes

- struct __gnu_pbds::container_error
- struct __gnu_pbds::insert_error
- struct __gnu_pbds::join_error
- struct __gnu_pbds::resize_error

Namespaces

- namespace __gnu_pbds

Functions

- void **__gnu_pbds::__throw_container_error** ()
- void **__gnu_pbds::__throw_insert_error** ()
- void **__gnu_pbds::__throw_join_error** ()
- void **__gnu_pbds::__throw_resize_error** ()

7.545.1 Detailed Description

Contains exception classes.

7.546 hash_policy.hpp File Reference

Classes

- class __gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >
- class __gnu_pbds::direct_mask_range_hashing< Size_Type >
- class __gnu_pbds::direct_mod_range_hashing< Size_Type >
- class __gnu_pbds::hash_exponential_size_policy< Size_Type >
- class __gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >
- class __gnu_pbds::hash_prime_size_policy
- class __gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >
- class __gnu_pbds::linear_probe_fn< Size_Type >
- class __gnu_pbds::quadratic_probe_fn< Size_Type >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_ASSERT_VALID**(X)
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_SIZE_BASE_C_DEC**

Enumerations

- enum { **num_distinct_sizes_16_bit** , **num_distinct_sizes_32_bit** , **num_distinct_sizes_64_bit** , **num_distinct_sizes** }

Variables

- static const std::size_t **__gnu_pbds::detail::g_a_sizes** [num_distinct_sizes]

7.546.1 Detailed Description

Contains hash-related policies.

7.547 list_update_policy.hpp File Reference

Classes

- class __gnu_pbds::lu_counter_policy< Max_Count, _Alloc >
- class __gnu_pbds::lu_move_to_front_policy< _Alloc >

Namespaces

- namespace __gnu_pbds

7.547.1 Detailed Description

Contains policies for list update containers.

7.548 priority_queue.hpp File Reference

Classes

- class __gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >

Namespaces

- namespace __gnu_pbds

7.548.1 Detailed Description

Contains priority_queues.

7.549 tag_and_trait.hpp File Reference

Classes

- struct __gnu_pbds::associative_tag
- struct __gnu_pbds::basic_branch_tag
- struct __gnu_pbds::basic_hash_tag
- struct __gnu_pbds::basic_invalidation_guarantee
- struct __gnu_pbds::binary_heap_tag
- struct __gnu_pbds::binomial_heap_tag
- struct __gnu_pbds::cc_hash_tag
- struct __gnu_pbds::container_tag
- struct __gnu_pbds::container_traits< Cntnr >
- struct __gnu_pbds::container_traits_base< binary_heap_tag >
- struct __gnu_pbds::container_traits_base< binomial_heap_tag >
- struct __gnu_pbds::container_traits_base< cc_hash_tag >
- struct __gnu_pbds::container_traits_base< gp_hash_tag >
- struct __gnu_pbds::container_traits_base< list_update_tag >
- struct __gnu_pbds::container_traits_base< ov_tree_tag >
- struct __gnu_pbds::container_traits_base< pairing_heap_tag >
- struct __gnu_pbds::container_traits_base< pat_trie_tag >
- struct __gnu_pbds::container_traits_base< rb_tree_tag >

- struct __gnu_pbds::container_traits_base< rc_binomial_heap_tag >
- struct __gnu_pbds::container_traits_base< splay_tree_tag >
- struct __gnu_pbds::container_traits_base< thin_heap_tag >
- struct __gnu_pbds::gp_hash_tag
- struct __gnu_pbds::list_update_tag
- struct __gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >
- struct __gnu_pbds::null_type
- struct __gnu_pbds::ov_tree_tag
- struct __gnu_pbds::pairing_heap_tag
- struct __gnu_pbds::pat_trie_tag
- struct __gnu_pbds::point_invalidation_guarantee
- struct __gnu_pbds::priority_queue_tag
- struct __gnu_pbds::range_invalidation_guarantee
- struct __gnu_pbds::rb_tree_tag
- struct __gnu_pbds::rc_binomial_heap_tag
- struct __gnu_pbds::sequence_tag
- struct __gnu_pbds::splay_tree_tag
- struct __gnu_pbds::string_tag
- struct __gnu_pbds::thin_heap_tag
- struct __gnu_pbds::tree_tag
- struct __gnu_pbds::trie_tag
- struct __gnu_pbds::trivial_iterator_tag

Namespaces

- namespace __gnu_pbds

Typedefs

- typedef void __gnu_pbds::trivial_iterator_difference_type

7.549.1 Detailed Description

Contains tags and traits, e.g., ones describing underlying data structures.

7.550 tree_policy.hpp File Reference

Classes

- class __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_BRANCH_POLICY_BASE**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**

7.550.1 Detailed Description

Contains tree-related policies.

7.551 trie_policy.hpp File Reference

Classes

- class __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >
- class __gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >
- struct __gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >

Namespaces

- namespace __gnu_pbds

Macros

- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_C_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_CLASS_T_DEC**
- #define **PB_DS_TRIE_POLICY_BASE**

Variables

- template<typename String, typename String::value_type Min_E_Val, typename String::value_type Max_E_Val, bool Reverse, typename _Alloc>
  detail::integral_constant< int, Reverse > **__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >::s_rev_ind**

7.551.1 Detailed Description

Contains trie-related policies.

7.552 pod_char_traits.h File Reference

Classes

- struct std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >
- struct __gnu_cxx::character< _Value, _Int, _St >

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- template<typename _Value, typename _Int, typename _St>
  bool **__gnu_cxx::operator<** (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)
- template<typename _Value, typename _Int, typename _St>
  bool **__gnu_cxx::operator==** (const character< _Value, _Int, _St > &lhs, const character< _Value, _Int, _St > &rhs)

7.552.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.553 pointer.h File Reference

Classes

- struct __gnu_cxx::_Invalid_type
- class __gnu_cxx::_Pointer_adapter< _Storage_policy >
- class __gnu_cxx::_Relative_pointer_impl< _Tp >
- class __gnu_cxx::_Relative_pointer_impl< const _Tp >
- class __gnu_cxx::_Std_pointer_impl< _Tp >
- struct __gnu_cxx::_Unqualified_type< _Tp >

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_CXX_POINTER_ARITH_OPERATOR_SET**(INT_TYPE)
- #define **_GCC_CXX_POINTER_COMPARISON_OPERATION_SET**(OPERATOR)

Functions

- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator!=** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator!=** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator!=** (const _Pointer_adapter< _Tp > &__lhs, int __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator!=** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator!=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &_↩
  _rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator!=** (int __lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator**< (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator**< (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator**< (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &_↩
  _rhs)
- template<typename _CharT, typename _Traits, typename _StoreT>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator**<< (std::basic_ostream< _CharT, _Traits > &↩
  __os, const _Pointer_adapter< _StoreT > &__p)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator**<= (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator**<= (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__↩
  rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator**<= (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)

- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator<=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &↩ __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator==** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp>
  bool __gnu_cxx::operator== (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator==** (const _Pointer_adapter< _Tp > &__lhs, int __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator==** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator==** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &↩ __rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator==** (int __lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator>** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator>** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator>** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator>** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &_↩ _rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator>=** (_Tp1 __lhs, const _Pointer_adapter< _Tp2 > &__rhs)
- template<typename _Tp>
  bool **__gnu_cxx::operator>=** (const _Pointer_adapter< _Tp > &__lhs, const _Pointer_adapter< _Tp > &__↩ rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator>=** (const _Pointer_adapter< _Tp1 > &__lhs, _Tp2 __rhs)
- template<typename _Tp1, typename _Tp2>
  bool **__gnu_cxx::operator>=** (const _Pointer_adapter< _Tp1 > &__lhs, const _Pointer_adapter< _Tp2 > &↩ __rhs)

7.553.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

Author

Bob Walters

Provides reusable _Pointer_adapter for assisting in the development of custom pointer types that can be used with the standard containers via the allocator::pointer and allocator::const_pointer typedefs.

7.554 pool_allocator.h File Reference

Classes

- class __gnu_cxx::__pool_alloc< _Tp >
- class __gnu_cxx::__pool_alloc_base

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _Tp>
  bool **__gnu_cxx::operator==** (const __pool_alloc< _Tp > &, const __pool_alloc< _Tp > &)

Variables

- template<typename _Tp>
  _Atomic_word **__gnu_cxx::__pool_alloc< _Tp >::_S_force_new**

7.554.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.555 rb_tree File Reference

Classes

- struct __gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >

Namespaces

- namespace __gnu_cxx

Macros

- #define **_RB_TREE**

7.555.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

7.556 rc_string_base.h File Reference

Classes

- class __gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >

Namespaces

- namespace __gnu_cxx

Variables

- template<typename _CharT, typename _Traits, typename _Alloc>
  __rc_string_base< _CharT, _Traits, _Alloc >::_Rep_empty **__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >::_S_empty_rep**

7.556.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

7.557 rope File Reference

Classes

- class __gnu_cxx::rope< _CharT, _Alloc >

Namespaces

- namespace __gnu_cxx
- namespace __gnu_cxx::__detail
- namespace std
- namespace std::tr1

Macros

- #define **__GC_CONST**
- #define **__ROPE_DEFINE_ALLOC**(_Tp, __name)
- #define **__ROPE_DEFINE_ALLOC**(_Tp, __name)
- #define **__ROPE_DEFINE_ALLOCS**(__a)
- #define **__STATIC_IF_SGI_ALLOC**
- #define **__STL_FREE_STRING**(__s, __l, __a)
- #define **__STL_ROPE_FROM_UNOWNED_CHAR_PTR**(__s, __size, __a)
- #define **_ROPE**

Typedefs

- typedef rope< char > **__gnu_cxx::crope**
- typedef rope< wchar_t > **__gnu_cxx::wrope**

Enumerations

- enum { **_S_max_rope_depth** }
- enum **_Tag** { **_S_leaf** , **_S_concat** , **_S_substringfn** , **_S_function** }

Functions

- crope::reference **__gnu_cxx::__mutable_reference_at** (crope &__c, std::size_t __i)
- template<typename _ForwardIterator, typename _Allocator>
  void **__gnu_cxx::_Destroy_const** (_ForwardIterator __first, _ForwardIterator __last, _Allocator __alloc)
- template<typename _ForwardIterator, typename _Tp>
  void **__gnu_cxx::_Destroy_const** (_ForwardIterator __first, _ForwardIterator __last, std::allocator< _Tp >)
- template<class _CharT>
  void **__gnu_cxx::_S_cond_store_eos** (_CharT &)
- void **__gnu_cxx::_S_cond_store_eos** (char &__c)
- void **__gnu_cxx::_S_cond_store_eos** (wchar_t &__c)
- template<class _CharT>
  _CharT **__gnu_cxx::_S_eos** (_CharT *)
- template<class _CharT>
  bool **__gnu_cxx::_S_is_basic_char_type** (_CharT *)
- bool **__gnu_cxx::_S_is_basic_char_type** (char *)
- bool **__gnu_cxx::_S_is_basic_char_type** (wchar_t *)
- template<class _CharT>
  bool **__gnu_cxx::_S_is_one_byte_char_type** (_CharT *)
- bool **__gnu_cxx::_S_is_one_byte_char_type** (char *)

- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator!=** (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr_↩ proxy< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator!=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_↩ iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator!=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator!=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  _Rope_const_iterator< _CharT, _Alloc > **__gnu_cxx::operator+** (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)
- template<class _CharT, class _Alloc>
  _Rope_iterator< _CharT, _Alloc > **__gnu_cxx::operator+** (const _Rope_iterator< _CharT, _Alloc > &__↩ x, std::ptrdiff_t __n)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **__gnu_cxx::operator+** (const rope< _CharT, _Alloc > &__left, _CharT __right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **__gnu_cxx::operator+** (const rope< _CharT, _Alloc > &__left, const _CharT *__right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **__gnu_cxx::operator+** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT, class _Alloc>
  _Rope_const_iterator< _CharT, _Alloc > **__gnu_cxx::operator+** (std::ptrdiff_t __n, const _Rope_const_↩ iterator< _CharT, _Alloc > &__x)
- template<class _CharT, class _Alloc>
  _Rope_iterator< _CharT, _Alloc > **__gnu_cxx::operator+** (std::ptrdiff_t __n, const _Rope_iterator< _CharT, _Alloc > &__x)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > & **__gnu_cxx::operator+=** (rope< _CharT, _Alloc > &__left, _CharT __right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > & **__gnu_cxx::operator+=** (rope< _CharT, _Alloc > &__left, const _CharT *__right)
- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > & **__gnu_cxx::operator+=** (rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT, class _Alloc>
  std::ptrdiff_t **__gnu_cxx::operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const↩ _iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  _Rope_const_iterator< _CharT, _Alloc > **__gnu_cxx::operator-** (const _Rope_const_iterator< _CharT, _Alloc > &__x, std::ptrdiff_t __n)
- template<class _CharT, class _Alloc>
  std::ptrdiff_t **__gnu_cxx::operator-** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _↩ CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  _Rope_iterator< _CharT, _Alloc > **__gnu_cxx::operator-** (const _Rope_iterator< _CharT, _Alloc > &__↩ x, std::ptrdiff_t __n)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator<** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_↩ iterator< _CharT, _Alloc > &__y)

- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator<** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator<** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT, class _Traits, class _Alloc>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator<<** (std::basic_ostream< _CharT, _Traits > &↩ __o, const rope< _CharT, _Alloc > &__r)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator<=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_↩ iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator<=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator<=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator==** (const _Rope_char_ptr_proxy< _CharT, _Alloc > &__x, const _Rope_char_ptr↩ _proxy< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator==** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_↩ iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator==** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator==** (const rope< _CharT, _Alloc > &__left, const rope< _CharT, _Alloc > &__right)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator>** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_↩ iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator>** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator>** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator>=** (const _Rope_const_iterator< _CharT, _Alloc > &__x, const _Rope_const_↩ iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator>=** (const _Rope_iterator< _CharT, _Alloc > &__x, const _Rope_iterator< _CharT, _Alloc > &__y)
- template<class _CharT, class _Alloc>
  bool **__gnu_cxx::operator>=** (const rope< _CharT, _Alloc > &__x, const rope< _CharT, _Alloc > &__y)
- template<class _CharT, class __Alloc>
  void **__gnu_cxx::swap** (_Rope_char_ref_proxy< _CharT, __Alloc > __a, _Rope_char_ref_proxy< _CharT, _↩ _Alloc > __b)
- template<class _CharT, class _Alloc>
  void **__gnu_cxx::swap** (rope< _CharT, _Alloc > &__x, rope< _CharT, _Alloc > &__y)

Variables

- template<class _CharT, class _Alloc>
  rope< _CharT, _Alloc > **__gnu_cxx::identity_element** (_Rope_Concat_fn< _CharT, _Alloc >)

- template<class _CharT, class _Alloc>
  const rope< _CharT, _Alloc >::size_type **__gnu_cxx::rope< _CharT, _Alloc >::npos**

7.557.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

7.558 ropeimpl.h File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **__GLIBCXX__**
- #define **__glibcxx_assert**(cond)
- #define **__N**(msgid)
- #define **__NO_CTYPE**
- #define **_GLIBCXX11_DEPRECATED**
- #define **_GLIBCXX11_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX11_USE_C99_MATH**
- #define **_GLIBCXX11_USE_C99_STDIO**
- #define **_GLIBCXX11_USE_C99_STDLIB**
- #define **_GLIBCXX11_USE_C99_WCHAR**
- #define **_GLIBCXX14_DEPRECATED**
- #define **_GLIBCXX14_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX17_DEPRECATED**
- #define **_GLIBCXX17_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX20_DEPRECATED**
- #define **_GLIBCXX20_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX23_DEPRECATED**
- #define **_GLIBCXX23_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX26_DEPRECATED**
- #define **_GLIBCXX26_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX98_USE_C99_COMPLEX**
- #define **_GLIBCXX98_USE_C99_MATH**
- #define **_GLIBCXX98_USE_C99_STDIO**
- #define **_GLIBCXX98_USE_C99_STDLIB**
- #define **_GLIBCXX98_USE_C99_WCHAR**
- #define **_GLIBCXX_ASSERT_FAIL**(_Condition)
- #define **_GLIBCXX_ATOMIC_WORD_BUILTINS**
- #define **_GLIBCXX_AUTO_CAST**(X)
- #define **_GLIBCXX_BEGIN_EXTERN_C**
- #define **_GLIBCXX_BEGIN_INLINE_ABI_NAMESPACE**(X)
- #define **_GLIBCXX_BEGIN_NAMESPACE_ALGO**
- #define **_GLIBCXX_BEGIN_NAMESPACE_CONTAINER**
- #define **_GLIBCXX_BEGIN_NAMESPACE_CXX11**
- #define **_GLIBCXX_BEGIN_NAMESPACE_LDBL**
- #define **_GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11**
- #define **_GLIBCXX_BEGIN_NAMESPACE_VERSION**
- #define **_GLIBCXX_CAN_ALIGNAS_DESTRUCTIVE_SIZE**

- #define **_GLIBCXX_CPU_DEFINES**
- #define **_GLIBCXX_DEFAULT_ABI_TAG**
- #define **_GLIBCXX_DEPRECATED**
- #define **_GLIBCXX_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX_DOXYGEN_ONLY**(X)
- #define **_GLIBCXX_END_EXTERN_C**
- #define **_GLIBCXX_END_INLINE_ABI_NAMESPACE**(X)
- #define **_GLIBCXX_END_NAMESPACE_ALGO**
- #define **_GLIBCXX_END_NAMESPACE_CONTAINER**
- #define **_GLIBCXX_END_NAMESPACE_CXX11**
- #define **_GLIBCXX_END_NAMESPACE_LDBL**
- #define **_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11**
- #define **_GLIBCXX_END_NAMESPACE_VERSION**
- #define **_GLIBCXX_EXTERN_TEMPLATE**
- #define **_GLIBCXX_EXTERN_TEMPLATE**
- #define **_GLIBCXX_FAST_MATH**
- #define **_GLIBCXX_FULLY_DYNAMIC_STRING**
- #define **_GLIBCXX_HAS_BUILTIN**(B)
- #define **_GLIBCXX_HAVE___CXA_THREAD_ATEXIT_IMPL**
- #define **_GLIBCXX_HAVE_ACOSF**
- #define **_GLIBCXX_HAVE_ACOSL**
- #define **_GLIBCXX_HAVE_ALIGNED_ALLOC**
- #define **_GLIBCXX_HAVE_ARC4RANDOM**
- #define **_GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE**
- #define **_GLIBCXX_HAVE_ASINF**
- #define **_GLIBCXX_HAVE_ASINL**
- #define **_GLIBCXX_HAVE_AT_QUICK_EXIT**
- #define **_GLIBCXX_HAVE_ATAN2F**
- #define **_GLIBCXX_HAVE_ATAN2L**
- #define **_GLIBCXX_HAVE_ATANF**
- #define **_GLIBCXX_HAVE_ATANL**
- #define **_GLIBCXX_HAVE_ATOMIC_LOCK_POLICY**
- #define **_GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY**
- #define **_GLIBCXX_HAVE_BUILTIN_HAS_UNIQ_OBJ_REP**
- #define **_GLIBCXX_HAVE_BUILTIN_IS_AGGREGATE**
- #define **_GLIBCXX_HAVE_BUILTIN_LAUNDER**
- #define **_GLIBCXX_HAVE_C99_FLT_EVAL_TYPES**
- #define **_GLIBCXX_HAVE_CEILF**
- #define **_GLIBCXX_HAVE_CEILL**
- #define **_GLIBCXX_HAVE_COSF**
- #define **_GLIBCXX_HAVE_COSHF**
- #define **_GLIBCXX_HAVE_COSHL**
- #define **_GLIBCXX_HAVE_COSL**
- #define **_GLIBCXX_HAVE_DECL_STRNLEN**
- #define **_GLIBCXX_HAVE_DIRFD**
- #define **_GLIBCXX_HAVE_EXCEPTION_PTR_SINCE_GCC46**
- #define **_GLIBCXX_HAVE_EXPF**
- #define **_GLIBCXX_HAVE_EXPL**
- #define **_GLIBCXX_HAVE_FABSF**
- #define **_GLIBCXX_HAVE_FABSL**
- #define **_GLIBCXX_HAVE_FDOPENDIR**

- #define **_GLIBCXX_HAVE_FINITE**
- #define **_GLIBCXX_HAVE_FINITEF**
- #define **_GLIBCXX_HAVE_FINITEL**
- #define **_GLIBCXX_HAVE_FLOORF**
- #define **_GLIBCXX_HAVE_FLOORL**
- #define **_GLIBCXX_HAVE_FMODF**
- #define **_GLIBCXX_HAVE_FMODL**
- #define **_GLIBCXX_HAVE_FREXPF**
- #define **_GLIBCXX_HAVE_FREXPL**
- #define **_GLIBCXX_HAVE_FWRITE_UNLOCKED**
- #define **_GLIBCXX_HAVE_GETENTROPY**
- #define **_GLIBCXX_HAVE_GETIPINFO**
- #define **_GLIBCXX_HAVE_GETS**
- #define **_GLIBCXX_HAVE_HYPOT**
- #define **_GLIBCXX_HAVE_HYPOTF**
- #define **_GLIBCXX_HAVE_HYPOTL**
- #define **_GLIBCXX_HAVE_ICONV**
- #define **_GLIBCXX_HAVE_IS_CONSTANT_EVALUATED**
- #define **_GLIBCXX_HAVE_ISINF**
- #define **_GLIBCXX_HAVE_ISINFF**
- #define **_GLIBCXX_HAVE_ISINFL**
- #define **_GLIBCXX_HAVE_ISNAN**
- #define **_GLIBCXX_HAVE_ISNANF**
- #define **_GLIBCXX_HAVE_ISNANL**
- #define **_GLIBCXX_HAVE_ISWBLANK**
- #define **_GLIBCXX_HAVE_LC_MESSAGES**
- #define **_GLIBCXX_HAVE_LDEXPF**
- #define **_GLIBCXX_HAVE_LDEXPL**
- #define **_GLIBCXX_HAVE_LIMIT_AS**
- #define **_GLIBCXX_HAVE_LIMIT_DATA**
- #define **_GLIBCXX_HAVE_LIMIT_FSIZE**
- #define **_GLIBCXX_HAVE_LIMIT_RSS**
- #define **_GLIBCXX_HAVE_LIMIT_VMEM**
- #define **_GLIBCXX_HAVE_LINK**
- #define **_GLIBCXX_HAVE_LINUX_FUTEX**
- #define **_GLIBCXX_HAVE_LOG10F**
- #define **_GLIBCXX_HAVE_LOG10L**
- #define **_GLIBCXX_HAVE_LOGF**
- #define **_GLIBCXX_HAVE_LOGL**
- #define **_GLIBCXX_HAVE_LSEEK**
- #define **_GLIBCXX_HAVE_MBSTATE_T**
- #define **_GLIBCXX_HAVE_MEMALIGN**
- #define **_GLIBCXX_HAVE_MODFF**
- #define **_GLIBCXX_HAVE_MODFL**
- #define **_GLIBCXX_HAVE_O_NONBLOCK**
- #define **_GLIBCXX_HAVE_OPENAT**
- #define **_GLIBCXX_HAVE_POLL**
- #define **_GLIBCXX_HAVE_POSIX_MEMALIGN**
- #define **_GLIBCXX_HAVE_POWF**
- #define **_GLIBCXX_HAVE_POWL**
- #define **_GLIBCXX_HAVE_QUICK_EXIT**

- #define **_GLIBCXX_HAVE_READLINK**
- #define **_GLIBCXX_HAVE_S_ISREG**
- #define **_GLIBCXX_HAVE_SECURE_GETENV**
- #define **_GLIBCXX_HAVE_SETENV**
- #define **_GLIBCXX_HAVE_SINCOS**
- #define **_GLIBCXX_HAVE_SINCOSF**
- #define **_GLIBCXX_HAVE_SINCOSL**
- #define **_GLIBCXX_HAVE_SINF**
- #define **_GLIBCXX_HAVE_SINHF**
- #define **_GLIBCXX_HAVE_SINHL**
- #define **_GLIBCXX_HAVE_SINL**
- #define **_GLIBCXX_HAVE_SOCKATMARK**
- #define **_GLIBCXX_HAVE_SQRTF**
- #define **_GLIBCXX_HAVE_SQRTL**
- #define **_GLIBCXX_HAVE_STACKTRACE**
- #define **_GLIBCXX_HAVE_STRERROR_L**
- #define **_GLIBCXX_HAVE_STRERROR_R**
- #define **_GLIBCXX_HAVE_STRTOF**
- #define **_GLIBCXX_HAVE_STRTOLD**
- #define **_GLIBCXX_HAVE_STRUCT_DIRENT_D_TYPE**
- #define **_GLIBCXX_HAVE_STRXFRM_L**
- #define **_GLIBCXX_HAVE_SYMLINK**
- #define **_GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT**
- #define **_GLIBCXX_HAVE_TANF**
- #define **_GLIBCXX_HAVE_TANHF**
- #define **_GLIBCXX_HAVE_TANHL**
- #define **_GLIBCXX_HAVE_TANL**
- #define **_GLIBCXX_HAVE_TIMESPEC_GET**
- #define **_GLIBCXX_HAVE_TLS**
- #define **_GLIBCXX_HAVE_TRUNCATE**
- #define **_GLIBCXX_HAVE_UNLINKAT**
- #define **_GLIBCXX_HAVE_USELOCALE**
- #define **_GLIBCXX_HAVE_VFWSCANF**
- #define **_GLIBCXX_HAVE_VSWSCANF**
- #define **_GLIBCXX_HAVE_VWSCANF**
- #define **_GLIBCXX_HAVE_WCSTOF**
- #define **_GLIBCXX_HAVE_WRITEV**
- #define **_GLIBCXX_HOSTED**
- #define **_GLIBCXX_ICONV_CONST**
- #define **_GLIBCXX_INLINE_VERSION**
- #define **_GLIBCXX_LT_OBJDIR**
- #define **_GLIBCXX_MANGLE_SIZE_T**
- #define **_GLIBCXX_MAY_HAVE___CXA_THREAD_ATEXIT_IMPL**
- #define **_GLIBCXX_NAMESPACE_CXX11**
- #define **_GLIBCXX_NAMESPACE_LDBL**
- #define **_GLIBCXX_NAMESPACE_LDBL_OR_CXX11**
- #define **_GLIBCXX_NO_OBSOLETE_ISINF_ISNAN_DYNAMIC**
- #define **_GLIBCXX_NO_SPECIALIZATIONS**
- #define **_GLIBCXX_NODISCARD**
- #define **_GLIBCXX_NOEXCEPT_PARM**
- #define **_GLIBCXX_NOEXCEPT_QUAL**

- #define **_GLIBCXX_OS_DEFINES**
- #define **_GLIBCXX_RELEASE**
- #define **_GLIBCXX_RES_LIMITS**
- #define **_GLIBCXX_STATIC_TZDATA**
- #define **_GLIBCXX_STD_A**
- #define **_GLIBCXX_STD_C**
- #define **_GLIBCXX_STDIO_EOF**
- #define **_GLIBCXX_STDIO_SEEK_CUR**
- #define **_GLIBCXX_STDIO_SEEK_END**
- #define **_GLIBCXX_SYMVER**
- #define **_GLIBCXX_SYMVER_GNU**
- #define **_GLIBCXX_TXN_SAFE**
- #define **_GLIBCXX_TXN_SAFE_DYN**
- #define **_GLIBCXX_USE_ALLOCATOR_NEW**
- #define **_GLIBCXX_USE_BUILTIN_TRAIT**(BT)
- #define **_GLIBCXX_USE_C11_UCHAR_CXX11**
- #define **_GLIBCXX_USE_C99**
- #define **_GLIBCXX_USE_C99_COMPLEX_TR1**
- #define **_GLIBCXX_USE_C99_CTYPE**
- #define **_GLIBCXX_USE_C99_CTYPE_TR1**
- #define **_GLIBCXX_USE_C99_FENV**
- #define **_GLIBCXX_USE_C99_FENV_TR1**
- #define **_GLIBCXX_USE_C99_INTTYPES**
- #define **_GLIBCXX_USE_C99_INTTYPES_TR1**
- #define **_GLIBCXX_USE_C99_INTTYPES_WCHAR_T**
- #define **_GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1**
- #define **_GLIBCXX_USE_C99_MATH_FUNCS**
- #define **_GLIBCXX_USE_C99_MATH_TR1**
- #define **_GLIBCXX_USE_C99_STDINT**
- #define **_GLIBCXX_USE_C99_STDINT_TR1**
- #define **_GLIBCXX_USE_CHDIR**
- #define **_GLIBCXX_USE_CHMOD**
- #define **_GLIBCXX_USE_CLOCK_MONOTONIC**
- #define **_GLIBCXX_USE_CLOCK_REALTIME**
- #define **_GLIBCXX_USE_DECIMAL_FLOAT**
- #define **_GLIBCXX_USE_DEV_RANDOM**
- #define **_GLIBCXX_USE_DUAL_ABI**
- #define **_GLIBCXX_USE_FCHMOD**
- #define **_GLIBCXX_USE_FCHMODAT**
- #define **_GLIBCXX_USE_FSEEKO_FTELLO**
- #define **_GLIBCXX_USE_GET_NPROCS**
- #define **_GLIBCXX_USE_GETCWD**
- #define **_GLIBCXX_USE_GETTIMEOFDAY**
- #define **_GLIBCXX_USE_GLIBC_STDIO_EXT**
- #define **_GLIBCXX_USE_INIT_PRIORITY_ATTRIBUTE**
- #define **_GLIBCXX_USE_LFS**
- #define **_GLIBCXX_USE_LONG_LONG**
- #define **_GLIBCXX_USE_LSTAT**
- #define **_GLIBCXX_USE_MKDIR**
- #define **_GLIBCXX_USE_NANOSLEEP**
- #define **_GLIBCXX_USE_NL_LANGINFO_L**

- #define **_GLIBCXX_USE_NLS**
- #define **_GLIBCXX_USE_PROC_SELF_STATUS**
- #define **_GLIBCXX_USE_PTHREAD_COND_CLOCKWAIT**
- #define **_GLIBCXX_USE_PTHREAD_MUTEX_CLOCKLOCK**
- #define **_GLIBCXX_USE_PTHREAD_RWLOCK_CLOCKLOCK**
- #define **_GLIBCXX_USE_RANDOM_TR1**
- #define **_GLIBCXX_USE_REALPATH**
- #define **_GLIBCXX_USE_SC_NPROCESSORS_ONLN**
- #define **_GLIBCXX_USE_SCHED_YIELD**
- #define **_GLIBCXX_USE_SENDFILE**
- #define **_GLIBCXX_USE_ST_MTIM**
- #define **_GLIBCXX_USE_STD_SPEC_FUNCS**
- #define **_GLIBCXX_USE_STDIO_LOCKING**
- #define **_GLIBCXX_USE_STRUCT_TM_TM_ZONE**
- #define **_GLIBCXX_USE_TMPNAM**
- #define **_GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_CXX20**
- #define **_GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_FCHAR8_T**
- #define **_GLIBCXX_USE_UTIME**
- #define **_GLIBCXX_USE_UTIMENSAT**
- #define **_GLIBCXX_USE_WCHAR_T**
- #define **_GLIBCXX_VERBOSE**
- #define **_GLIBCXX_VERBOSE_ASSERT**
- #define **_GLIBCXX_VISIBILITY**(V)
- #define **_GLIBCXX_X86_RDRAND**
- #define **_GLIBCXX_X86_RDSEED**
- #define **_GLIBCXX_ZONEINFO_DIR**
- #define **_GTHREAD_USE_MUTEX_TIMEDLOCK**

Functions

- template<class _CharT, class _Traits>
  void **__gnu_cxx::_Rope_fill** (std::basic_ostream< _CharT, _Traits > &__o, std::size_t __n)
- template<class _CharT>
  bool **__gnu_cxx::_Rope_is_simple** (_CharT *)
- bool **__gnu_cxx::_Rope_is_simple** (char *)
- bool **__gnu_cxx::_Rope_is_simple** (wchar_t *)
- template<class _Rope_iterator>
  void **__gnu_cxx::_Rope_rotate** (_Rope_iterator __first, _Rope_iterator __middle, _Rope_iterator __last)
- template<class _CharT, class _Traits, class _Alloc>
  std::basic_ostream< _CharT, _Traits > & **__gnu_cxx::operator**<< (std::basic_ostream< _CharT, _Traits > &↩ __o, const rope< _CharT, _Alloc > &__r)
- void **__gnu_cxx::rotate** (_Rope_iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __first, _Rope_↩ iterator< char, __STL_DEFAULT_ALLOCATOR(char)> __middle, _Rope_iterator< char, __STL_DEFAULT_↩ ALLOCATOR(char)> __last)

Variables

- template<class _CharT, class _Alloc>
  _CharT **__gnu_cxx::rope< _CharT, _Alloc >::_S_empty_c_str** [1]
- template<class _CharT, class _Alloc>
  const unsigned long **__gnu_cxx::rope< _CharT, _Alloc >::_S_min_len** [int(__detail::_S_max_rope_↩ depth)+1]

7.558.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/rope>.

7.559 slist File Reference

Classes

- class __gnu_cxx::slist< _Tp, _Alloc >

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **_SLIST**

Functions

- _Slist_node_base * **__gnu_cxx::__slist_make_link** (_Slist_node_base *__prev_node, _Slist_node_base *_↩ _new_node)
- _Slist_node_base * **__gnu_cxx::__slist_previous** (_Slist_node_base *__head, const _Slist_node_base *__↩ node)
- const _Slist_node_base * **__gnu_cxx::__slist_previous** (const _Slist_node_base *__head, const _Slist_↩ node_base *__node)
- _Slist_node_base * **__gnu_cxx::__slist_reverse** (_Slist_node_base *__node)
- std::size_t **__gnu_cxx::__slist_size** (_Slist_node_base *__node)
- void **__gnu_cxx::__slist_splice_after** (_Slist_node_base *__pos, _Slist_node_base *__before_first, _Slist_↩ node_base *__before_last)
- void **__gnu_cxx::__slist_splice_after** (_Slist_node_base *__pos, _Slist_node_base *__head)
- template<class _Tp, class _Alloc>
  bool **__gnu_cxx::operator!=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp, class _Alloc>
  bool **__gnu_cxx::operator<** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp, class _Alloc>
  bool **__gnu_cxx::operator<=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp, class _Alloc>
  bool **__gnu_cxx::operator==** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp, class _Alloc>
  bool **__gnu_cxx::operator>** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp, class _Alloc>
  bool **__gnu_cxx::operator>=** (const slist< _Tp, _Alloc > &_SL1, const slist< _Tp, _Alloc > &_SL2)
- template<class _Tp, class _Alloc>
  void **__gnu_cxx::swap** (slist< _Tp, _Alloc > &__x, slist< _Tp, _Alloc > &__y)

7.559.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

7.560 sso_string_base.h File Reference

Namespaces

- namespace __gnu_cxx

7.560.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

7.561 stdio_filebuf.h File Reference

Classes

- class __gnu_cxx::stdio_filebuf< _CharT, _Traits >

Namespaces

- namespace __gnu_cxx

7.561.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.562 stdio_sync_filebuf.h File Reference

Classes

- class __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >

Namespaces

- namespace __gnu_cxx

7.562.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.563 string_conversions.h File Reference

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _TRet, typename _Ret = _TRet, typename _CharT, typename... _Base>
  _Ret **__gnu_cxx::__stoa** (_TRet(*__convf)(const _CharT *, _CharT **, _Base...), const char *__name, const _CharT *__str, std::size_t *__idx, _Base... __base)
- template<typename _String, typename _CharT = typename _String::value_type>
  _String **__gnu_cxx::__to_xstring** (int(*__convf)(_CharT *, std::size_t, const _CharT *, __builtin_va_list), std::size_t __n, const _CharT *__fmt,...)

7.563.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.564 throw_allocator.h File Reference

Classes

- struct __gnu_cxx::limit_condition::always_adjustor
- struct __gnu_cxx::random_condition::always_adjustor
- struct __gnu_cxx::annotate_base
- struct __gnu_cxx::condition_base
- struct __gnu_cxx::forced_error
- struct __gnu_cxx::random_condition::group_adjustor
- struct std::hash< __gnu_cxx::throw_value_limit >
- struct std::hash< __gnu_cxx::throw_value_random >
- struct __gnu_cxx::limit_condition::limit_adjustor
- struct __gnu_cxx::limit_condition
- struct __gnu_cxx::limit_condition::never_adjustor
- struct __gnu_cxx::random_condition::never_adjustor
- struct __gnu_cxx::random_condition
- class __gnu_cxx::throw_allocator_base< _Tp, _Cond >
- struct __gnu_cxx::throw_allocator_limit< _Tp >
- struct __gnu_cxx::throw_allocator_random< _Tp >
- struct __gnu_cxx::throw_value_base< _Cond >
- struct __gnu_cxx::throw_value_limit
- struct __gnu_cxx::throw_value_random

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- void **__gnu_cxx::__throw_forced_error** ()
- template<typename _Cond>
  throw_value_base< _Cond > **__gnu_cxx::operator*** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond>
  throw_value_base< _Cond > **__gnu_cxx::operator+** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond>
  throw_value_base< _Cond > **__gnu_cxx::operator-** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond>
  bool **__gnu_cxx::operator<** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- std::ostream & **__gnu_cxx::operator<<** (std::ostream &os, const annotate_base &__b)
- template<typename _Tp, typename _Cond>
  bool **__gnu_cxx::operator==** (const throw_allocator_base< _Tp, _Cond > &, const throw_allocator_base< _Tp, _Cond > &)
- template<typename _Cond>
  bool **__gnu_cxx::operator==** (const throw_value_base< _Cond > &__a, const throw_value_base< _Cond > &__b)
- template<typename _Cond>
  void **__gnu_cxx::swap** (throw_value_base< _Cond > &__a, throw_value_base< _Cond > &__b)

7.564.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.
Contains two exception-generating types (throw_value, throw_allocator) intended to be used as value and allocator types while testing exception safety in templatized containers and algorithms. The allocator has additional log and debug features. The exception generated is of type forced_exception_error.

7.565 type_traits.h File Reference

Namespaces

- namespace __gnu_cxx

Functions

- template<typename _Type>
  constexpr bool **__gnu_cxx::__is_null_pointer** (_Type *__ptr)
- template<typename _Type>
  constexpr bool **__gnu_cxx::__is_null_pointer** (_Type)
- constexpr bool **__gnu_cxx::__is_null_pointer** (std::nullptr_t)

7.565.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.566 typelist.h File Reference

Namespaces

- namespace __gnu_cxx
- namespace __gnu_cxx::typelist

Macros

- #define **_GLIBCXX_TYPELIST_CHAIN1**(X0)
- #define **_GLIBCXX_TYPELIST_CHAIN10**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9)
- #define **_GLIBCXX_TYPELIST_CHAIN11**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10)
- #define **_GLIBCXX_TYPELIST_CHAIN12**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11)
- #define **_GLIBCXX_TYPELIST_CHAIN13**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12)
- #define **_GLIBCXX_TYPELIST_CHAIN14**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13)
- #define **_GLIBCXX_TYPELIST_CHAIN15**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14)
- #define **_GLIBCXX_TYPELIST_CHAIN16**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15)
- #define **_GLIBCXX_TYPELIST_CHAIN17**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16)
- #define **_GLIBCXX_TYPELIST_CHAIN18**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17)
- #define **_GLIBCXX_TYPELIST_CHAIN19**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18)
- #define **_GLIBCXX_TYPELIST_CHAIN2**(X0, X1)
- #define **_GLIBCXX_TYPELIST_CHAIN20**(X0, X1, X2, X3, X4, X5, X6, X7, X8, X9, X10, X11, X12, X13, X14, X15, X16, X17, X18, X19)
- #define **_GLIBCXX_TYPELIST_CHAIN3**(X0, X1, X2)
- #define **_GLIBCXX_TYPELIST_CHAIN4**(X0, X1, X2, X3)

- #define **_GLIBCXX_TYPELIST_CHAIN5**(X0, X1, X2, X3, X4)
- #define **_GLIBCXX_TYPELIST_CHAIN6**(X0, X1, X2, X3, X4, X5)
- #define **_GLIBCXX_TYPELIST_CHAIN7**(X0, X1, X2, X3, X4, X5, X6)
- #define **_GLIBCXX_TYPELIST_CHAIN8**(X0, X1, X2, X3, X4, X5, X6, X7)
- #define **_GLIBCXX_TYPELIST_CHAIN9**(X0, X1, X2, X3, X4, X5, X6, X7, X8)

Functions

- template<typename Fn, typename Typelist>
  void **__gnu_cxx::typelist::apply** (Fn &, Typelist)
- template<typename Fn, typename Typelist>
  void **__gnu_cxx::typelist::apply_generator** (Fn &fn, Typelist)
- template<typename Fn, typename TypelistT, typename TypelistV>
  void **__gnu_cxx::typelist::apply_generator** (Fn &fn, TypelistT, TypelistV)
- template<typename Gn, typename Typelist>
  void __gnu_cxx::typelist::apply_generator (Gn &, Typelist)
- template<typename Gn, typename TypelistT, typename TypelistV>
  void **__gnu_cxx::typelist::apply_generator** (Gn &, TypelistT, TypelistV)

7.566.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.
Contains typelist_chain definitions. Typelists are an idea by Andrei Alexandrescu.

7.567 vstring.h File Reference

Classes

- class __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >
- struct std::hash< __gnu_cxx::__u16vstring >
- struct std::hash< __gnu_cxx::__u32vstring >
- struct std::hash< __gnu_cxx::__vstring >
- struct std::hash< __gnu_cxx::__wvstring >

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator!= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *_↩
  _rhs)

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
bool __gnu_cxx::operator!= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_↩
_rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (__versa_string< _CharT, _Traits, ↩
_Alloc, _Base > &&__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (__versa_string< _CharT, _Traits, ↩
_Alloc, _Base > &&__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (__versa_string< _CharT, _Traits, ↩
_Alloc, _Base > &&__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (__versa_string< _CharT, _Traits, ↩
_Alloc, _Base > &&__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (_CharT __lhs, __versa_string< _↩
CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (const __versa_string< _CharT, _↩
Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > **__gnu_cxx::operator+** (const _CharT *__lhs, __versa_string< _CharT, _Traits, _Alloc, _Base > &&__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
__versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
bool __gnu_cxx::operator< (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *_↩
_rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
bool __gnu_cxx::operator< (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &_↩
_rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
basic_ostream< _CharT, _Traits > & std::operator<< (basic_ostream< _CharT, _Traits > &__os, const __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator<= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator<= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator== (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator== (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator> (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator> (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator>= (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  bool __gnu_cxx::operator>= (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)
- double **__gnu_cxx::stod** (const __vstring &__str, std::size_t *__idx=0)
- double **__gnu_cxx::stod** (const __wvstring &__str, std::size_t *__idx=0)
- float **__gnu_cxx::stof** (const __vstring &__str, std::size_t *__idx=0)
- float **__gnu_cxx::stof** (const __wvstring &__str, std::size_t *__idx=0)
- int **__gnu_cxx::stoi** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- int **__gnu_cxx::stoi** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- long **__gnu_cxx::stol** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- long **__gnu_cxx::stol** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- long double **__gnu_cxx::stold** (const __vstring &__str, std::size_t *__idx=0)
- long double **__gnu_cxx::stold** (const __wvstring &__str, std::size_t *__idx=0)
- long long **__gnu_cxx::stoll** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- long long **__gnu_cxx::stoll** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- unsigned long **__gnu_cxx::stoul** (const __vstring &__str, std::size_t *__idx=0, int __base=10)
- unsigned long **__gnu_cxx::stoul** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)
- unsigned long long **__gnu_cxx::stoull** (const __vstring &__str, std::size_t *__idx, int __base=10)
- unsigned long long **__gnu_cxx::stoull** (const __wvstring &__str, std::size_t *__idx=0, int __base=10)

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  void __gnu_cxx::swap (__versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, __versa_string< _CharT, _↩
  Traits, _Alloc, _Base > &__rhs)
- __vstring **__gnu_cxx::to_string** (double __val)
- __vstring **__gnu_cxx::to_string** (float __val)
- __vstring **__gnu_cxx::to_string** (int __val)
- __vstring **__gnu_cxx::to_string** (long __val)
- __vstring **__gnu_cxx::to_string** (long double __val)
- __vstring **__gnu_cxx::to_string** (long long __val)
- __vstring **__gnu_cxx::to_string** (unsigned __val)
- __vstring **__gnu_cxx::to_string** (unsigned long __val)
- __vstring **__gnu_cxx::to_string** (unsigned long long __val)
- __wvstring **__gnu_cxx::to_wstring** (double __val)
- __wvstring **__gnu_cxx::to_wstring** (float __val)
- __wvstring **__gnu_cxx::to_wstring** (int __val)
- __wvstring **__gnu_cxx::to_wstring** (long __val)
- __wvstring **__gnu_cxx::to_wstring** (long double __val)
- __wvstring **__gnu_cxx::to_wstring** (long long __val)
- __wvstring **__gnu_cxx::to_wstring** (unsigned __val)
- __wvstring **__gnu_cxx::to_wstring** (unsigned long __val)
- __wvstring **__gnu_cxx::to_wstring** (unsigned long long __val)

7.567.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.568 vstring.tcc File Reference

Namespaces

- namespace __gnu_cxx
- namespace std

Functions

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  basic_istream< _CharT, _Traits > & std::getline (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (_CharT __lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const __versa_string< _CharT, _Traits, _Alloc, _Base > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base>
  __versa_string< _CharT, _Traits, _Alloc, _Base > __gnu_cxx::operator+ (const _CharT *__lhs, const __versa_string< _CharT, _Traits, _Alloc, _Base > &__rhs)

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__is, __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base > &__str)

Variables

- template<typename _CharT, typename _Traits, typename _Alloc, template< typename, typename, typename > class _Base> const __versa_string< _CharT, _Traits, _Alloc, _Base >::size_type **__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >::npos**

7.568.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

7.569 vstring_fwd.h File Reference

Namespaces

- namespace __gnu_cxx

Typedefs

- typedef __versa_string< char, std::char_traits< char >, std::allocator< char >, __rc_string_base > **__gnu_cxx::__rc_string**
- typedef __vstring **__gnu_cxx::__sso_string**
- typedef __versa_string< char16_t, std::char_traits< char16_t >, std::allocator< char16_t >, __rc_string_base > **__gnu_cxx::__u16rc_string**
- typedef __u16vstring **__gnu_cxx::__u16sso_string**
- typedef __versa_string< char16_t > **__gnu_cxx::__u16vstring**
- typedef __versa_string< char32_t, std::char_traits< char32_t >, std::allocator< char32_t >, __rc_string_base > **__gnu_cxx::__u32rc_string**
- typedef __u32vstring **__gnu_cxx::__u32sso_string**
- typedef __versa_string< char32_t > **__gnu_cxx::__u32vstring**
- typedef __versa_string< char > **__gnu_cxx::__vstring**
- typedef __versa_string< wchar_t, std::char_traits< wchar_t >, std::allocator< wchar_t >, __rc_string_base > **__gnu_cxx::__wrc_string**
- typedef __wvstring **__gnu_cxx::__wsso_string**
- typedef __versa_string< wchar_t > **__gnu_cxx::__wvstring**

7.569.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

7.570 vstring_util.h File Reference

Namespaces

- namespace __gnu_cxx

7.570.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ext/vstring.h>.

7.571 fenv.h File Reference

7.571.1 Detailed Description

This is a Standard C++ Library header.

7.572 experimental/filesystem File Reference

Macros

- #define **__cpp_lib_experimental_filesystem**
- #define **_GLIBCXX_EXPERIMENTAL_FILESYSTEM**

7.572.1 Detailed Description

This is a TS C++ Library header.

7.573 filesystem File Reference

7.573.1 Detailed Description

This is a Standard C++ Library header.

7.574 flat_map File Reference

7.574.1 Detailed Description

This is a Standard C++ Library header.

7.575 flat_set File Reference

7.575.1 Detailed Description

This is a Standard C++ Library header.

7.576 format File Reference

7.576.1 Detailed Description

This is a Standard C++ Library header.

7.577 debug/forward_list File Reference

Classes

- class __gnu_debug::_Safe_forward_list< _SafeSequence >
- class std::__debug::forward_list< _Tp, _Alloc >

Namespaces

- namespace __gnu_debug
- namespace std
- namespace std::__debug

Macros

- #define **__glibcxx_check_valid_fl_range**(_First, _Last, _Dist)
- #define **_GLIBCXX20_ONLY**(__expr)
- #define **_GLIBCXX_DEBUG_FORWARD_LIST**
- #define **_GLIBCXX_FWDLIST_REMOVE_RETURN_TYPE_TAG**

Functions

- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::forward_list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> forward_list< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  std::__debug::forward_list (size_t, _Tp, _Allocator=_Allocator()) -> forward_list< _Tp, _Allocator >
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **std::__debug::operator<=>** (const forward_list< _Tp, _Alloc > &__x, const forward_list< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool **std::__debug::operator==** (const forward_list< _Tp, _Alloc > &__lx, const forward_list< _Tp, _Alloc > &__ly)
- template<typename _Tp, typename _Alloc>
  void std::__debug::swap (forward_list< _Tp, _Alloc > &__lx, forward_list< _Tp, _Alloc > &__ly) noexcept(noexcept(__lx.swap(__ly)))

7.577.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.578 experimental/forward_list File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_FORWARD_LIST**

Typedefs

- template<typename _Tp>
  using **std::experimental::fundamentals_v2::pmr::forward_list**

Functions

- template<typename _Tp, typename _Alloc, typename _Up>
  void **std::experimental::erase** (forward_list< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (forward_list< _Tp, _Alloc > &__cont, _Predicate __pred)

7.578.1 Detailed Description

This is a TS C++ Library header.

7.579 forward_list File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Tp>
  using **std::pmr::forward_list**

7.579.1 Detailed Description

This is a Standard C++ Library header.

7.580 fstream File Reference

Classes

- class std::basic_filebuf< _CharT, _Traits >
- class std::basic_fstream< _CharT, _Traits >
- class std::basic_ifstream< _CharT, _Traits >
- class std::basic_ofstream< _CharT, _Traits >

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_BUFSIZ**

Typedefs

- template<typename _Path, typename _Result = _Path, typename _Path2 = decltype(std::declval<_Path&>().make_preferred().filename())>
  using **std::_If_fs_path**

Functions

- template<class _CharT, class _Traits>
  void std::swap (basic_filebuf< _CharT, _Traits > &__x, basic_filebuf< _CharT, _Traits > &__y)
- template<class _CharT, class _Traits>
  void std::swap (basic_fstream< _CharT, _Traits > &__x, basic_fstream< _CharT, _Traits > &__y)
- template<class _CharT, class _Traits>
  void std::swap (basic_ifstream< _CharT, _Traits > &__x, basic_ifstream< _CharT, _Traits > &__y)
- template<class _CharT, class _Traits>
  void std::swap (basic_ofstream< _CharT, _Traits > &__x, basic_ofstream< _CharT, _Traits > &__y)

7.580.1 Detailed Description

This is a Standard C++ Library header.

7.581 experimental/functional File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_boyer_moore_searching**
- #define **__cpp_lib_experimental_not_fn**
- #define **_GLIBCXX_EXPERIMENTAL_FUNCTIONAL**

Typedefs

- template<typename _RAIter, typename _Hash, typename _Pred, typename _Val = typename iterator_traits<_RAIter>::value_type, typename _Diff = typename iterator_traits<_RAIter>::difference_type>
  using **std::experimental::__boyer_moore_base_t**

Functions

- template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _BinaryPredicate = equal_to<>>
  boyer_moore_horspool_searcher< _RAIter, _Hash, _BinaryPredicate > std::experimental::make_boyer_moore_horspool_searcher (_RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())
- template<typename _RAIter, typename _Hash = std::hash<typename std::iterator_traits<_RAIter>::value_type>, typename _BinaryPredicate = equal_to<>>
  boyer_moore_searcher< _RAIter, _Hash, _BinaryPredicate > std::experimental::make_boyer_moore_searcher (_RAIter __pat_first, _RAIter __pat_last, _Hash __hf=_Hash(), _BinaryPredicate __pred=_BinaryPredicate())
- template<typename _ForwardIterator, typename _BinaryPredicate = std::equal_to<>>
  default_searcher< _ForwardIterator, _BinaryPredicate > std::experimental::make_default_searcher (_ForwardIterator __pat_first, _ForwardIterator __pat_last, _BinaryPredicate __pred=_BinaryPredicate())
- template<typename _Fn>
  auto std::experimental::not_fn (_Fn &&__fn) noexcept(std::is_nothrow_constructible< std::decay_t< _Fn >, _Fn && >::value)

Variables

- template<typename _Tp>
  constexpr bool std::experimental::is_bind_expression_v
- template<typename _Tp>
  constexpr int std::experimental::is_placeholder_v

7.581.1 Detailed Description

This is a TS C++ Library header.

7.582 ext/functional File Reference

Classes

- class __gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >
- struct __gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >
- struct __gnu_cxx::constant_unary_fun< _Result, _Argument >
- struct __gnu_cxx::constant_void_fun< _Result >

- struct __gnu_cxx::project1st< _Arg1, _Arg2 >
- struct __gnu_cxx::project2nd< _Arg1, _Arg2 >
- struct __gnu_cxx::select1st< _Pair >
- struct __gnu_cxx::select2nd< _Pair >
- class __gnu_cxx::subtractive_rng
- class __gnu_cxx::unary_compose< _Operation1, _Operation2 >

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_FUNCTIONAL**

Functions

- template<class _Operation1, class _Operation2>
  unary_compose< _Operation1, _Operation2 > __gnu_cxx::compose1 (const _Operation1 &__fn1, const _↩
  Operation2 &__fn2)
- template<class _Operation1, class _Operation2, class _Operation3>
  binary_compose< _Operation1, _Operation2, _Operation3 > __gnu_cxx::compose2 (const _Operation1 &__fn1, const _Operation2 &__fn2, const _Operation3 &__fn3)
- template<class _Result>
  constant_void_fun< _Result > __gnu_cxx::constant0 (const _Result &__val)
- template<class _Result>
  constant_unary_fun< _Result, _Result > __gnu_cxx::constant1 (const _Result &__val)
- template<class _Result>
  constant_binary_fun< _Result, _Result, _Result > __gnu_cxx::constant2 (const _Result &__val)
- template<class _Tp>
  _Tp __gnu_cxx::identity_element (std::multiplies< _Tp >)
- template<class _Tp>
  _Tp __gnu_cxx::identity_element (std::plus< _Tp >)
- template<class _Ret, class _Tp, class _Arg>
  std::const_mem_fun1_t< _Ret, _Tp, _Arg > **__gnu_cxx::mem_fun1** (_Ret(_Tp::*__f)(_Arg) const)
- template<class _Ret, class _Tp, class _Arg>
  std::mem_fun1_t< _Ret, _Tp, _Arg > **__gnu_cxx::mem_fun1** (_Ret(_Tp::*__f)(_Arg))
- template<class _Ret, class _Tp, class _Arg>
  std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg > **__gnu_cxx::mem_fun1_ref** (_Ret(_Tp::*__f)(_Arg) const)
- template<class _Ret, class _Tp, class _Arg>
  std::mem_fun1_ref_t< _Ret, _Tp, _Arg > **__gnu_cxx::mem_fun1_ref** (_Ret(_Tp::*__f)(_Arg))

7.582.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

7.583 functional File Reference

Classes

- class std::_Not_fn< _Fn >
- struct std::_Placeholder< _Num >
- struct std::is_bind_expression< _Tp >
- struct std::is_bind_expression< _Bind< _Signature > >

- struct std::is_bind_expression< _Bind_result< _Result, _Signature > >
- struct std::is_bind_expression< const _Bind< _Signature > >
- struct std::is_bind_expression< const _Bind_result< _Result, _Signature > >
- struct std::is_bind_expression< const volatile _Bind< _Signature > >
- struct std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >
- struct std::is_bind_expression< volatile _Bind< _Signature > >
- struct std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >
- struct std::is_placeholder< _Tp >
- struct std::is_placeholder< _Placeholder< _Num > >

Namespaces

- namespace std
- namespace std::placeholders

Macros

- #define **_GLIBCXX_NOT_FN_CALL_OP**(_QUALS)
- #define **_GLIBCXX_PLACEHOLDER**

Typedefs

- template<typename _Tp, typename _Tp2 = typename decay<_Tp>::type>
  using **std::__is_socketlike**

Functions

- template<typename _Result, typename _Func, typename... _BoundArgs>
  constexpr _Bindres_helper< _Result, _Func, _BoundArgs... >::type std::bind (_Func &&__f, _BoundArgs &&... __args)
- template<typename _Func, typename... _BoundArgs>
  constexpr _Bind_helper< __is_socketlike< _Func >::value, _Func, _BoundArgs... >::type std::bind (_Func &&↩ __f, _BoundArgs &&... __args)
- template<typename _Tp, typename _Class>
  constexpr _Mem_fn< _Tp _Class::* > std::mem_fn (_Tp _Class::*__pm) noexcept

Variables

- const _Placeholder< 1 > **std::placeholders::_1**
- const _Placeholder< 10 > **std::placeholders::_10**
- const _Placeholder< 11 > **std::placeholders::_11**
- const _Placeholder< 12 > **std::placeholders::_12**
- const _Placeholder< 13 > **std::placeholders::_13**
- const _Placeholder< 14 > **std::placeholders::_14**
- const _Placeholder< 15 > **std::placeholders::_15**
- const _Placeholder< 16 > **std::placeholders::_16**
- const _Placeholder< 17 > **std::placeholders::_17**
- const _Placeholder< 18 > **std::placeholders::_18**
- const _Placeholder< 19 > **std::placeholders::_19**
- const _Placeholder< 2 > **std::placeholders::_2**
- const _Placeholder< 20 > **std::placeholders::_20**
- const _Placeholder< 21 > **std::placeholders::_21**
- const _Placeholder< 22 > **std::placeholders::_22**

- const _Placeholder< 23 > **std::placeholders::_23**
- const _Placeholder< 24 > **std::placeholders::_24**
- const _Placeholder< 25 > **std::placeholders::_25**
- const _Placeholder< 26 > **std::placeholders::_26**
- const _Placeholder< 27 > **std::placeholders::_27**
- const _Placeholder< 28 > **std::placeholders::_28**
- const _Placeholder< 29 > **std::placeholders::_29**
- const _Placeholder< 3 > **std::placeholders::_3**
- const _Placeholder< 4 > **std::placeholders::_4**
- const _Placeholder< 5 > **std::placeholders::_5**
- const _Placeholder< 6 > **std::placeholders::_6**
- const _Placeholder< 7 > **std::placeholders::_7**
- const _Placeholder< 8 > **std::placeholders::_8**
- const _Placeholder< 9 > **std::placeholders::_9**
- template<typename _Tp>
  constexpr bool **std::is_bind_expression_v**
- template<typename _Tp>
  constexpr int **std::is_placeholder_v**

7.583.1 Detailed Description

This is a Standard C++ Library header.

7.584 future File Reference

Classes

- class std::__basic_future< _Res >
- class std::future< _Res >
- class std::future< _Res & >
- class std::future< void >
- class std::future_error
- struct std::is_error_code_enum< future_errc >
- class std::packaged_task< _Res(_ArgTypes...)>
- class std::promise< _Res >
- class std::promise< _Res & >
- class std::promise< void >
- class std::shared_future< _Res >
- class std::shared_future< _Res & >
- class std::shared_future< void >

Namespaces

- namespace std

Enumerations

- enum class std::future_errc { **future_already_retrieved** , **promise_already_satisfied** , **no_state** , **broken_promise** }
- enum class std::future_status { **ready** , **timeout** , **deferred** }
- enum class std::launch { **async** , **deferred** }

Functions

- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > std::async (_Fn &&__fn, _Args &&... __args)
- template<typename _Fn, typename... _Args>
  future< __async_result_of< _Fn, _Args... > > std::async (launch __policy, _Fn &&__fn, _Args &&... __args)
- const error_category & std::future_category () noexcept
- error_code std::make_error_code (future_errc __errc) noexcept
- error_condition std::make_error_condition (future_errc __errc) noexcept
- constexpr launch **std::operator&** (launch __x, launch __y) noexcept
- constexpr launch & **std::operator&=** (launch &__x, launch __y) noexcept
- constexpr launch **std::operator^** (launch __x, launch __y) noexcept
- constexpr launch & **std::operator^=** (launch &__x, launch __y) noexcept
- constexpr launch **std::operator|** (launch __x, launch __y) noexcept
- constexpr launch & **std::operator|=** (launch &__x, launch __y) noexcept
- constexpr launch **std::operator~** (launch __x) noexcept
- template<typename _Fun, typename _Signature = __function_guide_t<_Fun, decltype(&_Fun::operator())>>
  std::packaged_task (_Fun) -> packaged_task< _Signature >
- template<typename _Res, typename... _ArgTypes>
  std::packaged_task (_Res(*)(_ArgTypes...)) -> packaged_task< _Res(_ArgTypes...)>
- template<typename _Res, typename... _ArgTypes>
  void std::swap (packaged_task< _Res(_ArgTypes...)> &__x, packaged_task< _Res(_ArgTypes...)> &__y) noexcept
- template<typename _Res>
  void **std::swap** (promise< _Res > &__x, promise< _Res > &__y) noexcept

7.584.1 Detailed Description

This is a Standard C++ Library header.

7.585 generator File Reference

7.585.1 Detailed Description

This is a Standard C++ Library header.

7.586 inplace_vector File Reference

Namespaces

- namespace std

Functions

- template<typename _Tp, size_t _Nm, typename _Up = _Tp>
  constexpr size_t **std::erase** (::inplace_vector< _Tp, _Nm > &__cont, const _Up &__value)
- template<typename _Tp, size_t _Nm, typename _Predicate>
  constexpr size_t **std::erase_if** (::inplace_vector< _Tp, _Nm > &__cont, _Predicate __pred)

7.586.1 Detailed Description

This is a Standard C++ Library header.

7.587 iomanip File Reference

Namespaces

- namespace std

Functions

- template<typename _MoneyT>
  _Get_money< _MoneyT > std::get_money (_MoneyT &__mon, bool __intl=false)
- template<typename _CharT>
  _Get_time< _CharT > std::get_time (std::tm *__tmb, const _CharT *__fmt)
- template<typename _CharT, typename _Traits, typename _MoneyT>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Put_money< _MoneyT > __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Put_time< _CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Resetiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setbase __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setfill< _CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setprecision __f)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator**<< (basic_ostream< _CharT, _Traits > &__os, _Setw __f)
- template ostream & **std::operator**<< (ostream &, _Resetiosflags)
- template ostream & **std::operator**<< (ostream &, _Setbase)
- template ostream & **std::operator**<< (ostream &, _Setfill< char >)
- template ostream & **std::operator**<< (ostream &, _Setiosflags)
- template ostream & **std::operator**<< (ostream &, _Setprecision)
- template ostream & **std::operator**<< (ostream &, _Setw)
- template wostream & **std::operator**<< (wostream &, _Resetiosflags)
- template wostream & **std::operator**<< (wostream &, _Setbase)
- template wostream & **std::operator**<< (wostream &, _Setfill< wchar_t >)
- template wostream & **std::operator**<< (wostream &, _Setiosflags)
- template wostream & **std::operator**<< (wostream &, _Setprecision)
- template wostream & **std::operator**<< (wostream &, _Setw)
- template<typename _CharT, typename _Traits, typename _MoneyT>
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Get_money< _MoneyT > __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator**>> (basic_istream< _CharT, _Traits > &__is, _Get_time< _CharT > __f)

- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator>>** (basic_istream< _CharT, _Traits > &__is, _Resetiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator>>** (basic_istream< _CharT, _Traits > &__is, _Setbase __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator>>** (basic_istream< _CharT, _Traits > &__is, _Setfill< _↩
  CharT > __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator>>** (basic_istream< _CharT, _Traits > &__is, _Setiosflags __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator>>** (basic_istream< _CharT, _Traits > &__is, _Setprecision __f)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & **std::operator>>** (basic_istream< _CharT, _Traits > &__is, _Setw __f)
- template istream & **std::operator>>** (istream &, _Resetiosflags)
- template istream & **std::operator>>** (istream &, _Setbase)
- template istream & **std::operator>>** (istream &, _Setfill< char >)
- template istream & **std::operator>>** (istream &, _Setiosflags)
- template istream & **std::operator>>** (istream &, _Setprecision)
- template istream & **std::operator>>** (istream &, _Setw)
- template wistream & **std::operator>>** (wistream &, _Resetiosflags)
- template wistream & **std::operator>>** (wistream &, _Setbase)
- template wistream & **std::operator>>** (wistream &, _Setfill< wchar_t >)
- template wistream & **std::operator>>** (wistream &, _Setiosflags)
- template wistream & **std::operator>>** (wistream &, _Setprecision)
- template wistream & **std::operator>>** (wistream &, _Setw)
- template<typename _MoneyT>
  _Put_money< _MoneyT > std::put_money (const _MoneyT &__mon, bool __intl=false)
- template<typename _CharT>
  _Put_time< _CharT > std::put_time (const std::tm *__tmb, const _CharT *__fmt)
- _Resetiosflags std::resetiosflags (ios_base::fmtflags __mask)
- _Setbase std::setbase (int __base)
- template<typename _CharT>
  _Setfill< _CharT > std::setfill (_CharT __c)
- _Setiosflags std::setiosflags (ios_base::fmtflags __mask)
- _Setprecision std::setprecision (int __n)
- _Setw std::setw (int __n)

7.587.1 Detailed Description

This is a Standard C++ Library header.

7.588 ios File Reference

7.588.1 Detailed Description

This is a Standard C++ Library header.

7.589 iosfwd File Reference

7.589.1 Detailed Description

This is a Standard C++ Library header.
This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

7.590 iostream File Reference

Namespaces

- namespace std

Variables

- static ios_base::Init **std::__ioinit**

Standard Stream Objects

The <iostream> header declares the eight standard stream objects. For other declarations, see https://gcc.gnu.org/onlinedocs/libstdc++/manual/io.html and the I/O forward declarations

They are required by default to cooperate with the global C library's FILE streams, and to be available during program startup and termination. For more information, see the section of the manual linked to above.

- ostream std::cerr
- istream std::cin
- ostream std::clog
- ostream std::cout
- wostream std::wcerr
- wistream std::wcin
- wostream std::wclog
- wostream std::wcout

7.590.1 Detailed Description

This is a Standard C++ Library header.

7.591 istream File Reference

Classes

- class std::basic_iostream< _CharT, _Traits >
- class std::basic_istream< _CharT, _Traits >
- class std::basic_istream< _CharT, _Traits >::sentry

Namespaces

- namespace std

Typedefs

- template<typename _Is, typename _Tp>
  using **std::__rvalue_stream_extraction_t**

Functions

- template<typename _CharT, typename _Traits>
  void **std::__istream_extract** (basic_istream< _CharT, _Traits > &, _CharT *, streamsize)
- void **std::__istream_extract** (istream &, char *, streamsize)
- template<typename _Istream, typename _Tp>
  __rvalue_stream_extraction_t< _Istream, _Tp > std::operator>> (_Istream &&__is, _Tp &&__x)
- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & std::ws (basic_istream< _CharT, _Traits > &__is)

- template<typename _CharT, typename _Traits>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT &__c)
- template<class _Traits>
  basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char &__c)
- template<class _Traits>
  basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char &__c)

- template<typename _CharT, typename _Traits, size_t _Num>
  basic_istream< _CharT, _Traits > & std::operator>> (basic_istream< _CharT, _Traits > &__in, _CharT(&__s)[_Num])
- template<class _Traits, size_t _Num>
  basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, signed char(&__s)[_Num])
- template<class _Traits, size_t _Num>
  basic_istream< char, _Traits > & std::operator>> (basic_istream< char, _Traits > &__in, unsigned char(&__s)[_Num])

7.591.1 Detailed Description

This is a Standard C++ Library header.

7.592 experimental/iterator File Reference

Classes

- class std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits >

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_ostream_joiner**
- #define **_GLIBCXX_EXPERIMENTAL_ITERATOR**

Functions

- template<typename _CharT, typename _Traits, typename _DelimT>
  ostream_joiner< decay_t< _DelimT >, _CharT, _Traits > std::experimental::make_ostream_joiner (basic_ostream< _CharT, _Traits > &__os, _DelimT &&__delimiter)

7.592.1 Detailed Description

This is a TS C++ Library header.

7.593 ext/iterator File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_ITERATOR**

Functions

- template<typename _InputIterator, typename _Distance>
  void **__gnu_cxx::__distance** (_InputIterator __first, _InputIterator __last, _Distance &__n, std::input_iterator_tag)
- template<typename _RandomAccessIterator, typename _Distance>
  void **__gnu_cxx::__distance** (_RandomAccessIterator __first, _RandomAccessIterator __last, _Distance &__n, std::random_access_iterator_tag)
- template<typename _InputIterator, typename _Distance>
  void __gnu_cxx::distance (_InputIterator __first, _InputIterator __last, _Distance &__n)

7.593.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

7.594 iterator File Reference

7.594.1 Detailed Description

This is a Standard C++ Library header.

7.595 latch File Reference

7.595.1 Detailed Description

This is a Standard C++ Library header.

7.596 limits File Reference

Classes

- struct std::__numeric_limits_base
- struct std::numeric_limits< _Tp >
- struct std::numeric_limits< bool >
- struct std::numeric_limits< char >
- struct std::numeric_limits< char16_t >
- struct std::numeric_limits< char32_t >
- struct std::numeric_limits< double >
- struct std::numeric_limits< float >
- struct std::numeric_limits< int >
- struct std::numeric_limits< long >
- struct std::numeric_limits< long double >
- struct std::numeric_limits< long long >

- struct std::numeric_limits< short >
- struct std::numeric_limits< signed char >
- struct std::numeric_limits< unsigned char >
- struct std::numeric_limits< unsigned int >
- struct std::numeric_limits< unsigned long >
- struct std::numeric_limits< unsigned long long >
- struct std::numeric_limits< unsigned short >
- struct std::numeric_limits< wchar_t >

Namespaces

- namespace std

Macros

- #define **__glibcxx_concat3**(P, M, S)
- #define **__glibcxx_concat3_**(P, M, S)
- #define **__glibcxx_digits**(T)
- #define **__glibcxx_digits10**(T)
- #define **__glibcxx_digits10_b**(T, B)
- #define **__glibcxx_digits_b**(T, B)
- #define **__glibcxx_double_has_denorm_loss**
- #define **__glibcxx_double_tinyness_before**
- #define **__glibcxx_double_traps**
- #define **__glibcxx_float_has_denorm_loss**
- #define **__glibcxx_float_n**(BITSIZE)
- #define **__glibcxx_float_tinyness_before**
- #define **__glibcxx_float_traps**
- #define **__glibcxx_integral_traps**
- #define **__glibcxx_long_double_has_denorm_loss**
- #define **__glibcxx_long_double_tinyness_before**
- #define **__glibcxx_long_double_traps**
- #define **__glibcxx_max**(T)
- #define **__glibcxx_max_b**(T, B)
- #define **__glibcxx_max_digits10**(T)
- #define **__glibcxx_min**(T)
- #define **__glibcxx_min_b**(T, B)
- #define **__glibcxx_signed**(T)
- #define **__glibcxx_signed_b**(T, B)
- #define **__INT_N**(TYPE, BITSIZE, EXT, UEXT)
- #define **__INT_N_201103**(TYPE)
- #define **__INT_N_U201103**(TYPE)
- #define **__max_digits10**

Enumerations

- enum std::float_denorm_style { std::denorm_indeterminate , std::denorm_absent , std::denorm_present }
- enum std::float_round_style { std::round_indeterminate , std::round_toward_zero , std::round_to_nearest , std::round_toward_infinity , std::round_toward_neg_infinity }

7.596.1 Detailed Description

This is a Standard C++ Library header.

7.597 debug/list File Reference

Classes

- class std::__debug::list< _Tp, _Allocator >

Namespaces

- namespace __gnu_debug
- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX20_ONLY**(__expr)
- #define **_GLIBCXX_DEBUG_LIST**
- #define **_GLIBCXX_LIST_REMOVE_RETURN_TYPE_TAG**

Functions

- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::list (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> list< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  std::__debug::list (size_t, _Tp, _Allocator=_Allocator()) -> list< _Tp, _Allocator >
- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **std::__debug::operator<=>** (const list< _Tp, _Alloc > &__x, const list< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  bool **std::__debug::operator==** (const list< _Tp, _Alloc > &__lhs, const list< _Tp, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc>
  void **std::__debug::swap** (list< _Tp, _Alloc > &__lhs, list< _Tp, _Alloc > &__rhs) noexcept(/*conditional */)

7.597.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.598 experimental/list File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_LIST**

Typedefs

- template<typename _Tp>
  using **std::experimental::fundamentals_v2::pmr::list**

Functions

- template<typename _Tp, typename _Alloc, typename _Up>
  void **std::experimental::erase** (list< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (list< _Tp, _Alloc > &__cont, _Predicate __pred)

7.598.1 Detailed Description

This is a TS C++ Library header.

7.599 list File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Tp>
  using **std::pmr::list**

7.599.1 Detailed Description

This is a Standard C++ Library header.

7.600 locale File Reference

Macros

- #define **_GLIBCXX_LOCALE**

7.600.1 Detailed Description

This is a Standard C++ Library header.

7.601 debug/map File Reference

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_MAP**

7.601.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.602 experimental/map File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_MAP**

Typedefs

- template<typename _Key, typename _Tp, typename _Compare = less<_Key>>
  using **std::experimental::fundamentals_v2::pmr::map**
- template<typename _Key, typename _Tp, typename _Compare = less<_Key>>
  using **std::experimental::fundamentals_v2::pmr::multimap**

Functions

- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (map< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Compare, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (multimap< _Key, _Tp, _Compare, _Alloc > &__cont, _Predicate __pred)

7.602.1 Detailed Description

This is a TS C++ Library header.

7.603 map File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Key, typename _Tp, typename _Cmp = std::less<_Key>>
  using **std::pmr::map**
- template<typename _Key, typename _Tp, typename _Cmp = std::less<_Key>>
  using **std::pmr::multimap**

7.603.1 Detailed Description

This is a Standard C++ Library header.

7.604 math.h File Reference

Functions

- template<typename _Tp>
  _Tp abs (const complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **acos** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **acosh** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **asin** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **asinh** (const std::complex< _Tp > &)
- template<typename _Tp>
  std::complex< _Tp > **atan** (const std::complex< _Tp > &)

- constexpr float **atan2** (float __y, float __x)
- template<typename _Tp>
  std::complex< _Tp > **atanh** (const std::complex< _Tp > &)
- constexpr float **cbrt** (float __x)
- constexpr float **ceil** (float __x)
- constexpr float **copysign** (float __x, float __y)
- template<typename _Tp>
  complex< _Tp > cos (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > cosh (const complex< _Tp > &)
- constexpr float **erf** (float __x)
- constexpr float **erfc** (float __x)
- template<typename _Tp>
  complex< _Tp > exp (const complex< _Tp > &)
- constexpr float **exp2** (float __x)
- constexpr float **expm1** (float __x)
- template<typename _Tp>
  _Tp fabs (const std::complex< _Tp > &__z)
- constexpr float **fdim** (float __x, float __y)
- constexpr float **floor** (float __x)
- constexpr float **fma** (float __x, float __y, float __z)
- constexpr float **fmax** (float __x, float __y)
- constexpr float **fmin** (float __x, float __y)
- constexpr float **fmod** (float __x, float __y)
- constexpr int **fpclassify** (float __x)
- float **frexp** (float __x, int *__exp)
- constexpr float **hypot** (float __x, float __y)
- constexpr int **ilogb** (float __x)
- constexpr bool **isfinite** (float __x)
- constexpr bool **isgreater** (float __x, float __y)
- constexpr bool **isgreaterequal** (float __x, float __y)
- constexpr bool **isinf** (float __x)
- constexpr bool **isless** (float __x, float __y)
- constexpr bool **islessequal** (float __x, float __y)
- constexpr bool **islessgreater** (float __x, float __y)
- constexpr bool **isnan** (float __x)
- constexpr bool **isnormal** (float __x)
- constexpr bool **isunordered** (float __x, float __y)
- constexpr float **ldexp** (float __x, int __exp)
- constexpr float **lgamma** (float __x)
- constexpr long long **llrint** (float __x)
- constexpr long long **llround** (float __x)
- template<typename _Tp>
  complex< _Tp > log (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > log10 (const complex< _Tp > &)
- constexpr float **log1p** (float __x)
- constexpr float **log2** (float __x)
- constexpr float **logb** (float __x)
- constexpr long **lrint** (float __x)
- constexpr long **lround** (float __x)

- float **modf** (float __x, float *__iptr)
- constexpr float **nearbyint** (float __x)
- constexpr float **nextafter** (float __x, float __y)
- constexpr float **nexttoward** (float __x, long double __y)
- template<typename _Tp>
  complex< _Tp > pow (const complex< _Tp > &, int)
- constexpr float **remainder** (float __x, float __y)
- float **remquo** (float __x, float __y, int *__pquo)
- constexpr float **rint** (float __x)
- constexpr float **round** (float __x)
- constexpr float **scalbln** (float __x, long __ex)
- constexpr float **scalbn** (float __x, int __ex)
- constexpr bool **signbit** (float __x)
- template<typename _Tp>
  complex< _Tp > sin (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > sinh (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > sqrt (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > tan (const complex< _Tp > &)
- template<typename _Tp>
  complex< _Tp > tanh (const complex< _Tp > &)
- constexpr float **tgamma** (float __x)
- constexpr float **trunc** (float __x)

7.604.1 Detailed Description

This is a Standard C++ Library header.

7.604.2 Function Documentation

abs()

```
template<typename _Tp>
_Tp std::abs (
            const complex< _Tp > & __z)  [inline]
```

Return magnitude of *z*.

cos()

```
template<typename _Tp>
complex< _Tp > std::cos (
            const complex< _Tp > & __z)  [inline]
```

Return complex cosine of *z*.

cosh()

```
template<typename _Tp>
complex< _Tp > std::cosh (
            const complex< _Tp > & __z)  [inline]
```

Return complex hyperbolic cosine of *z*.

exp()

```
template<typename _Tp>
complex< _Tp > std::exp (
            const complex< _Tp > & __z)  [inline]
```

Return complex base e exponential of *z*.

fabs()

```
template<typename _Tp>
_Tp std::fabs (
            const std::complex< _Tp > & __z)  [inline]
```

fabs(__z) TR1 8.1.8 [tr.c99.cmplx.fabs]

log()

```
template<typename _Tp>
complex< _Tp > std::log (
            const complex< _Tp > & __z)  [inline]
```

Return complex natural logarithm of *z*.

log10()

```
template<typename _Tp>
complex< _Tp > std::log10 (
            const complex< _Tp > & __z)  [inline]
```

Return complex base 10 logarithm of *z*.

pow()

```
template<typename _Tp>
complex< _Tp > std::pow (
            const complex< _Tp > & __z,
            int __n)  [inline]
```

Return *x* to the *y'th* power.

sin()

```
template<typename _Tp>
complex< _Tp > std::sin (
            const complex< _Tp > & __z)  [inline]
```

Return complex sine of *z*.

sinh()

```
template<typename _Tp>
complex< _Tp > std::sinh (
            const complex< _Tp > & __z)  [inline]
```

Return complex hyperbolic sine of *z*.

sqrt()

```
template<typename _Tp>
complex< _Tp > std::sqrt (
            const complex< _Tp > & __z)  [inline]
```

Return complex square root of *z*.

tan()

```
template<typename _Tp>
complex< _Tp > std::tan (
            const complex< _Tp > & __z)  [inline]
```

Return complex tangent of *z*.

tanh()

```
template<typename _Tp>
complex< _Tp > std::tanh (
            const complex< _Tp > & __z)  [inline]
```

Return complex hyperbolic tangent of *z*.

7.605 mdspan File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Slice>
  using **std::__mdspan::__full_extent_t**
- template<typename _IndexType, size_t _Rank>
  using **std::dextents**
- template<size_t _Rank, typename _IndexType = size_t>
  using **std::dims**

Enumerations

- enum class **_LayoutSide** { **__left** , **__right** , **__unknown** }
- enum class **_SliceKind** { **__extent_slice** , **__unit_stride_slice** , **__full** , **__collapsing** }

Functions

- consteval bool **std::__mdspan::__all_dynamic** (std::span< const size_t > __extents)
- consteval bool **std::__mdspan::__all_static** (std::span< const size_t > __extents)
- template<typename _IndexType, size_t... _Counts>
  auto **std::__mdspan::__build_dextents_type** (integer_sequence< size_t, _Counts... >) -> extents< _Index↩
  Type,((void) _Counts, dynamic_extent)... >
- template<typename _IndexType, typename _Slice>
  constexpr auto **std::__mdspan::__canonical_index** (_Slice &&__slice)
- template<typename _IndexType, typename _OffsetType, typename _SpanType, typename _StrideType>
  constexpr auto **std::__mdspan::__canonical_range_slice** (_OffsetType __offset, _SpanType __span, _Stride↩
  Type __stride)
- template<typename _IndexType, size_t _Extent, typename _OIndexType>
  constexpr void **std::__mdspan::__check_inrange_index** (const extents< _IndexType, _Extent > &__ext, const
  _OIndexType &__idx)
- template<typename _IndexType, size_t _Extent, typename _OIndexType>
  constexpr void **std::__mdspan::__check_valid_index** (const extents< _IndexType, _Extent > &__ext, const
  _OIndexType &__idx)
- template<typename _IndexType, size_t _Extent, typename _Slice>
  constexpr void **std::__mdspan::__check_valid_slice** (const extents< _IndexType, _Extent > &__ext, const ↩
  _Slice &__slice)

- template<typename _Extents, typename... _Slices>
  constexpr void **std::__mdspan::__check_valid_slices** (const _Extents &__exts, const _Slices &... __slices)
- template<typename _Tp, size_t _Nm>
  consteval bool **std::__mdspan::__contains_zero** (const array< _Tp, _Nm > &__exts) noexcept
- template<typename _Tp, size_t _Nm>
  constexpr bool **std::__mdspan::__contains_zero** (span< _Tp, _Nm > __exts) noexcept
- template<typename _Extents>
  constexpr span< const typename _Extents::index_type > **std::__mdspan::__dynamic_extents** (const _Extents &__exts, size_t __begin=0, size_t __end=_Extents::rank()) noexcept
- template<size_t _K, typename _Extents, typename _Slice>
  constexpr _Extents::index_type **std::__mdspan::__dynamic_slice_extent** (const _Extents &__exts, _Slice _↩
  _slice)
- template<typename _Extents>
  constexpr bool **std::__mdspan::__empty** (const _Extents &__exts) noexcept
- template<typename _Extents>
  constexpr _Extents::index_type **std::__mdspan::__extents_prod** (const _Extents &__exts, size_t __sta_prod, size_t __begin, size_t __end) noexcept
- template<size_t _Index, typename _Extents>
  constexpr auto **std::__mdspan::__extract_extent** (const _Extents &__exts)
- template<typename _Extents>
  constexpr _Extents::index_type **std::__mdspan::__fwd_prod** (const _Extents &__exts, size_t __begin, size_t __end) noexcept
- template<typename _Extents>
  constexpr _Extents::index_type **std::__mdspan::__fwd_prod** (const _Extents &__exts, size_t __r) noexcept
- template<typename _IndexType, size_t _Nm>
  consteval _IndexType **std::__mdspan::__fwd_prod** (span< const _IndexType, _Nm > __values)
- template<typename _PaddedMapping>
  consteval size_t **std::__mdspan::__get_static_stride** ()
- template<typename _IndexType, typename _OIndexTypeRef>
  constexpr _IndexType **std::__mdspan::__index_type_cast** (_OIndexTypeRef &&__other)
- template<typename _IndexType, typename... _Slices>
  consteval auto **std::__mdspan::__inv_map_rank** ()
- consteval bool **std::__mdspan::__is_block** (span< const _SliceKind > __slice_kinds, size_t __block_size)
- template<typename _Extents>
  constexpr bool **std::__mdspan::__is_representable_extents** (const _Extents &__exts) noexcept
- template<typename _IndexType, typename _OIndexType>
  consteval bool **std::__mdspan::__is_representable_integer** (_OIndexType __value)
- template<typename _IndexType>
  constexpr bool **std::__mdspan::__is_representable_least_multiple** (size_t __x, size_t __y)
- template<size_t _PaddedStride, typename _Extents, typename _LayoutTraits>
  consteval bool **std::__mdspan::__is_representable_padded_size** ()
- constexpr size_t **std::__mdspan::__least_multiple** (size_t __x, size_t __y)
- template<typename _Extents, typename... _Indices>
  constexpr _Extents::index_type **std::__mdspan::__linear_index_left** (const _Extents &__exts, _Indices... __↩
  indices) noexcept
- template<typename _Extents, typename _Stride, typename... _Indices>
  constexpr _Extents::index_type **std::__mdspan::__linear_index_leftpad** (const _Extents &__exts, _Stride __↩
  stride, _Indices... __indices)
- template<typename _Extents, typename... _Indices>
  constexpr _Extents::index_type **std::__mdspan::__linear_index_right** (const _Extents &__exts, _Indices... _↩
  _indices) noexcept

- template<typename _Extents, typename _Stride, typename... _Indices>
  constexpr _Extents::index_type **std::__mdspan::__linear_index_rightpad** (const _Extents &__exts, _Stride ↩ __stride, _Indices... __indices)
- template<typename _Mapping, typename... _Indices>
  constexpr _Mapping::index_type **std::__mdspan::__linear_index_strides** (const _Mapping &__m, _Indices... __indices) noexcept
- template<typename _Slice>
  consteval _SliceKind **std::__mdspan::__make_slice_kind** ()
- template<typename... _Slices>
  consteval array< _SliceKind, sizeof...(_Slices)> **std::__mdspan::__make_slice_kind_array** ()
- template<typename _Mapping>
  consteval _LayoutSide **std::__mdspan::__mapping_side** ()
- template<typename _Mapping>
  constexpr _Mapping::index_type **std::__mdspan::__offset** (const _Mapping &__m) noexcept
- template<_LayoutSide _Side, size_t _Nm>
  static consteval size_t **std::__mdspan::__padded_block_begin** (span< const _SliceKind, _Nm > __slice_ ↩ kinds, size_t __sub_rank)
- static consteval size_t **std::__mdspan::__padded_block_begin_generic** (span< const _SliceKind > __slice ↩ _kinds, size_t __sub_rank)
- template<typename _Extents>
  constexpr _Extents::index_type **std::__mdspan::__rev_prod** (const _Extents &__exts, size_t __r) noexcept
- template<typename _Extents>
  constexpr _Extents::index_type **std::__mdspan::__size** (const _Extents &__exts) noexcept
- template<typename _Slice>
  constexpr auto **std::__mdspan::__slice_begin** (_Slice __slice)
- template<typename _IndexType, typename _Slice>
  constexpr auto **std::__mdspan::__slice_cast** (_Slice &&__slice)
- template<typename _Extents>
  constexpr const array< size_t, _Extents::rank()> & **std::__mdspan::__static_extents** () noexcept
- template<typename _Extents>
  constexpr span< const size_t > **std::__mdspan::__static_extents** (size_t __begin, size_t __end) noexcept
- template<typename _Extents, typename _IndexType = typename _Extents::index_type>
  consteval _IndexType **std::__mdspan::__static_quotient** (_IndexType __nom=__gnu_cxx::__int_traits< _ ↩ IndexType >::__max)
- template<typename _IndexType>
  consteval _IndexType **std::__mdspan::__static_quotient** (std::span< const size_t > __sta_exts, _IndexType __nom=__gnu_cxx::__int_traits< _IndexType >::__max)
- template<typename _IndexType, size_t _Extent, typename _Slice>
  consteval size_t **std::__mdspan::__static_slice_extent** ()
- template<typename _IndexType, size_t... _Extents, typename... _Slices>
  constexpr auto **std::__mdspan::__subextents** (const extents< _IndexType, _Extents... > &__exts, _Slices... __slices)
- template<typename _Mapping, typename... _Slices>
  constexpr auto **std::__mdspan::__submapping** (const _Mapping &__mapping, _Slices... __slices)
- template<typename _Mapping>
  constexpr auto **std::__mdspan::__submdspan_mapping_impl** (const _Mapping &__mapping)
- template<typename _Mapping, typename... _Slices>
  constexpr auto **std::__mdspan::__submdspan_mapping_impl** (const _Mapping &__mapping, _Slices... __ ↩ slices)
- template<typename _Mapping, typename... _Slices>
  constexpr size_t **std::__mdspan::__suboffset** (const _Mapping &__mapping, const _Slices &... __slices)
- template<typename _IndexType, typename... _Slices>
  consteval auto **std::__mdspan::__subrank** ()

- template<typename _SubExts, typename _Mapping, typename... _Slices>
  constexpr auto **std::__mdspan::__substrides** (const _Mapping &__mapping, const _Slices &... __slices)
- template<typename _SubExts, typename _Mapping, typename... _Slices>
  constexpr auto **std::__mdspan::__substrides_generic** (const _Mapping &__mapping, const _Slices &... __slices)
- template<typename _SubExts, typename _Mapping, typename... _Slices>
  constexpr auto **std::__mdspan::__substrides_standardized** (const _Mapping &__mapping, const _Slices &... __slices)
- template<typename _IndexType, size_t... _Extents, typename... _RawSlices>
  constexpr auto **std::canonical_slices** (const extents< _IndexType, _Extents... > &__exts, _RawSlices... __raw_slices)
- template<typename... _Integrals>
  std::extents (_Integrals...) -> extents< size_t, __detail::__maybe_static_ext< _Integrals >... >
- template<typename _CArray>
  std::mdspan (_CArray &) -> mdspan< remove_all_extents_t< _CArray >, extents< size_t, extent_v< _CArray, 0 > > >
- template<typename _ElementType, typename... _Integrals>
  std::mdspan (_ElementType *, _Integrals...) -> mdspan< _ElementType, extents< size_t, __detail::__maybe_static_ext< _Integrals >... > >
- template<typename _ElementType, typename _MappingType>
  std::mdspan (_ElementType *, const _MappingType &) -> mdspan< _ElementType, typename _MappingType::extents_type, typename _MappingType::layout_type >
- template<typename _ElementType, typename _OIndexType, size_t _Nm>
  std::mdspan (_ElementType *, const array< _OIndexType, _Nm > &) -> mdspan< _ElementType, dextents< size_t, _Nm > >
- template<typename _ElementType, typename _IndexType, size_t... _ExtentsPack>
  std::mdspan (_ElementType *, const extents< _IndexType, _ExtentsPack... > &) -> mdspan< _ElementType, extents< _IndexType, _ExtentsPack... > >
- template<typename _ElementType, typename _OIndexType, size_t _Nm>
  std::mdspan (_ElementType *, span< _OIndexType, _Nm >) -> mdspan< _ElementType, dextents< size_t, _Nm > >
- template<typename _Pointer>
  std::mdspan (_Pointer &&) -> mdspan< remove_pointer_t< remove_reference_t< _Pointer > >, extents< size_t > >
- template<typename _MappingType, typename _AccessorType>
  std::mdspan (typename _AccessorType::data_handle_type, const _MappingType &, const _AccessorType &) -> mdspan< typename _AccessorType::element_type, typename _MappingType::extents_type, typename _MappingType::layout_type, _AccessorType >
- template<typename _IndexType, size_t... _Extents, typename... _RawSlices>
  constexpr auto **std::subextents** (const extents< _IndexType, _Extents... > &__exts, _RawSlices... __raw_slices)
- template<typename _ElementType, typename _Extents, typename _Layout, typename _Accessor, typename... _RawSlices>
  constexpr auto **std::submdspan** (const mdspan< _ElementType, _Extents, _Layout, _Accessor > &__md, _RawSlices... __raw_slices)
- void **std::__mdspan::submdspan_mapping** ()=delete

Variables

- template<array _Extents>
  constexpr auto **std::__mdspan::__fwd_partial_prods**
- template<typename _Tp>
  constexpr bool **std::__mdspan::__is_extent_slice**

- template<typename _OffsetType, typename _ExtentType, typename _StrideType>
  constexpr bool **std::__mdspan::__is_extent_slice**< **extent_slice**< **_OffsetType, _ExtentType, _StrideType** > >
- template<typename _Tp>
  constexpr bool **std::__mdspan::__is_extents**
- template<typename _IndexType, size_t... _Extents>
  constexpr bool **std::__mdspan::__is_extents**< **extents**< **_IndexType, _Extents...** > >
- template<typename _Mapping>
  constexpr bool **std::__mdspan::__is_left_padded_mapping**
- template<typename _Mapping>
  constexpr bool **std::__mdspan::__is_padded_mapping**
- template<typename _Tp>
  constexpr bool **std::__mdspan::__is_range_slice**
- template<typename _FirstType, typename _LastType, typename _StrideType>
  constexpr bool **std::__mdspan::__is_range_slice**< **range_slice**< **_FirstType, _LastType, _StrideType** > >
- template<typename _Mapping>
  constexpr bool **std::__mdspan::__is_right_padded_mapping**
- template<typename _Tp>
  constexpr bool **std::__is_submdspan_mapping_result**
- template<typename _Mapping>
  constexpr bool **std::__is_submdspan_mapping_result**< **submdspan_mapping_result**< **_Mapping** > >
- template<array _Extents>
  constexpr auto **std::__mdspan::__rev_partial_prods**
- constexpr full_extent_t **std::full_extent**

7.605.1 Detailed Description

This is a Standard C++ Library header.

7.606 experimental/memory File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_observer_ptr**
- #define **_GLIBCXX_EXPERIMENTAL_MEMORY**

Functions

- template<typename _Tp>
  constexpr observer_ptr< _Tp > **std::experimental::make_observer** (_Tp *__p) noexcept
- template<typename _Tp>
  constexpr bool **std::experimental::operator!=** (nullptr_t, observer_ptr< _Tp > __p) noexcept
- template<typename _Tp>
  bool **std::experimental::operator!=** (observer_ptr< _Tp > __p, nullptr_t) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator!=** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator**< (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)

- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator<=** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _Tp>
  constexpr bool **std::experimental::operator==** (nullptr_t, observer_ptr< _Tp > __p) noexcept
- template<typename _Tp>
  constexpr bool **std::experimental::operator==** (observer_ptr< _Tp > __p, nullptr_t) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator==** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::operator>=** (observer_ptr< _Tp > __p1, observer_ptr< _Up > __p2)
- template<typename _Tp>
  constexpr void **std::experimental::swap** (observer_ptr< _Tp > &__p1, observer_ptr< _Tp > &__p2) noexcept

7.606.1 Detailed Description

This is a TS C++ Library header.

7.607 ext/memory File Reference

Classes

- class __gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >
- struct __gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_MEMORY**

Functions

- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n** (_InputIter __first, _Size __count, ↩
  _ForwardIter __result)
- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n** (_InputIter __first, _Size __count, ↩
  _ForwardIter __result, std::input_iterator_tag)
- template<typename _RandomAccessIter, typename _Size, typename _ForwardIter>
  std::pair< _RandomAccessIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n** (_RandomAccessIter _↩
  _first, _Size __count, _ForwardIter __result, std::random_access_iterator_tag)
- template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Allocator>
  std::pair< _InputIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n_a** (_InputIter __first, _Size __count, _ForwardIter __result, _Allocator __alloc)
- template<typename _InputIter, typename _Size, typename _ForwardIter, typename _Tp>
  std::pair< _InputIter, _ForwardIter > **__gnu_cxx::__uninitialized_copy_n_a** (_InputIter __first, _Size __count, _ForwardIter __result, std::allocator< _Tp >)
- template<typename _InputIter, typename _Size, typename _ForwardIter>
  std::pair< _InputIter, _ForwardIter > __gnu_cxx::uninitialized_copy_n (_InputIter __first, _Size __count, _↩
  ForwardIter __result)

7.607.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

7.608 memory File Reference

7.608.1 Detailed Description

This is a Standard C++ Library header.

7.609 experimental/memory_resource File Reference

Namespaces

- namespace __gnu_cxx
- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_memory_resources**
- #define **_GLIBCXX_EXPERIMENTAL_MEMORY_RESOURCE**
- #define **_GLIBCXX_MAX_ALIGN_MATCHES_MALLOC**

Typedefs

- template<typename _Alloc>
  using **std::experimental::fundamentals_v2::pmr::resource_adaptor**

Functions

- memory_resource * std::experimental::fundamentals_v2::pmr::get_default_resource () noexcept
- memory_resource * **std::experimental::fundamentals_v2::pmr::new_delete_resource** () noexcept
- memory_resource * **std::experimental::fundamentals_v2::pmr::null_memory_resource** () noexcept
- bool **std::experimental::fundamentals_v2::pmr::operator!=** (const memory_resource &__a, const memory_resource &__b) noexcept
- template<class _Tp1, class _Tp2>
  bool **std::experimental::fundamentals_v2::pmr::operator!=** (const polymorphic_allocator< _Tp1 > &__a, const polymorphic_allocator< _Tp2 > &__b) noexcept
- bool **std::experimental::fundamentals_v2::pmr::operator==** (const memory_resource &__a, const memory_resource &__b) noexcept
- template<class _Tp1, class _Tp2>
  bool **std::experimental::fundamentals_v2::pmr::operator==** (const polymorphic_allocator< _Tp1 > &__a, const polymorphic_allocator< _Tp2 > &__b) noexcept
- memory_resource * std::experimental::fundamentals_v2::pmr::set_default_resource (memory_resource *__r) noexcept

7.609.1 Detailed Description

This is a TS C++ Library header.

7.609.2 Function Documentation

get_default_resource()

```
memory_resource * std::experimental::fundamentals_v2::pmr::get_default_resource ()  [inline],
[noexcept]
```
Get the current default resource.

set_default_resource()

```
memory_resource * std::experimental::fundamentals_v2::pmr::set_default_resource (
            memory_resource * __r)  [inline], [noexcept]
```
Change the default resource and return the previous one.

7.610 memory_resource File Reference

Classes

- class std::pmr::monotonic_buffer_resource
- struct std::pmr::pool_options
- class std::pmr::unsynchronized_pool_resource

Namespaces

- namespace std

Functions

- memory_resource * std::pmr::get_default_resource () noexcept
- memory_resource * std::pmr::new_delete_resource () noexcept
- memory_resource * std::pmr::null_memory_resource () noexcept
- memory_resource * std::pmr::set_default_resource (memory_resource *__r) noexcept

7.610.1 Detailed Description

This is a Standard C++ Library header.
This header declares the pmr (std::pmr) memory resources.

7.610.2 Function Documentation

get_default_resource()

```
memory_resource * std::pmr::get_default_resource ()  [noexcept]
```
Get the current default memory resource pointer.

null_memory_resource()

```
memory_resource * std::pmr::null_memory_resource ()  [nodiscard], [noexcept]
```
A pmr::memory_resource that always throws `bad_alloc`.

set_default_resource()

```
memory_resource * std::pmr::set_default_resource (
            memory_resource * __r)  [noexcept]
```
Replace the default memory resource pointer.

7.611 mutex File Reference

Classes

- struct std::once_flag
- class std::recursive_mutex
- class std::recursive_timed_mutex
- class std::timed_mutex

Namespaces

- namespace std

Functions

- void **std::__once_proxy** (void)
- template<typename _Callable, typename... _Args>
  void std::call_once (once_flag &__once, _Callable &&__f, _Args &&... __args)
- template<typename _L1, typename _L2, typename... _L3>
  void std::lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)
- template<typename _L1, typename _L2, typename... _L3>
  int std::try_lock (_L1 &__l1, _L2 &__l2, _L3 &... __l3)

7.611.1 Detailed Description

This is a Standard C++ Library header.

7.612 numbers File Reference

7.612.1 Detailed Description

This is a Standard C++ Library header.

7.613 experimental/numeric File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_gcd_lcm**
- #define **_GLIBCXX_EXPERIMENTAL_NUMERIC**

Functions

- template<typename _Mn, typename _Nn>
  constexpr common_type_t< _Mn, _Nn > std::experimental::gcd (_Mn __m, _Nn __n) noexcept
- template<typename _Mn, typename _Nn>
  constexpr common_type_t< _Mn, _Nn > std::experimental::lcm (_Mn __m, _Nn __n)

7.613.1 Detailed Description

This is a TS C++ Library header.

7.614 ext/numeric File Reference

Namespaces

- namespace __gnu_cxx

Macros

- #define **_EXT_NUMERIC**

Functions

- template<typename _Tp, typename _Integer>
  _Tp **__gnu_cxx::__power** (_Tp __x, _Integer __n)
- template<typename _Tp, typename _Integer, typename _MonoidOperation>
  _Tp **__gnu_cxx::__power** (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)
- template<typename _ForwardIterator, typename _Tp>
  constexpr void __gnu_cxx::iota (_ForwardIterator __first, _ForwardIterator __last, _Tp __value)
- template<typename _Tp, typename _Integer>
  _Tp __gnu_cxx::power (_Tp __x, _Integer __n)
- template<typename _Tp, typename _Integer, typename _MonoidOperation>
  _Tp __gnu_cxx::power (_Tp __x, _Integer __n, _MonoidOperation __monoid_op)

7.614.1 Detailed Description

This file is a GNU extension to the Standard C++ Library (possibly containing extensions from the HP/SGI STL subset).

7.615 numeric File Reference

Namespaces

- namespace std
- namespace std::__detail

Macros

- #define **_PSTL_NUMERIC_FORWARD_DECLARED**

Typedefs

- template<typename _Out, typename _Tp>
  using **std::ranges::iota_result**

Functions

- template<typename _Res, typename _Tp>
  constexpr _Res **std::__detail::__abs_r** (_Tp __val)
- template<typename>
  void **std::__detail::__abs_r** (bool)=delete
- template<typename _Tp>
  constexpr _Tp **std::__detail::__gcd** (_Tp __m, _Tp __n)
- template<typename _InputIterator, typename _OutputIterator, typename _Tp>
  constexpr _OutputIterator std::exclusive_scan (_InputIterator __first, _InputIterator __last, _OutputIterator __result, _Tp __init)

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

Variables

- constexpr __iota_fn **std::ranges::iota**

7.615.1 Detailed Description

This is a Standard C++ Library header.

7.616 parallel/numeric File Reference

Namespaces

- namespace std
- namespace std::__parallel

Functions

- template<typename __RAIter, typename _Tp, typename _BinaryOperation>
  _Tp **std::__parallel::__accumulate_switch** (__RAIter __begin, __RAIter __end, _Tp __init, _BinaryOperation __binary_op, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOperation, typename _IteratorTag>
  _Tp **std::__parallel::__accumulate_switch** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, _IteratorTag)
- template<typename _IIter, typename _Tp, typename _IteratorTag>
  _Tp **std::__parallel::__accumulate_switch** (_IIter __begin, _IIter __end, _Tp __init, _IteratorTag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>
  _OutputIterator **std::__parallel::__adjacent_difference_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::__adjacent_difference_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _IteratorTag1, typename _IteratorTag2>
  _Tp **std::__parallel::__inner_product_switch** (_IIter1 __first1, _IIter1 __last1, _IIter2 __first2, _Tp __init, _BinaryFunction1 __binary_op1, _BinaryFunction2 __binary_op2, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::__inner_product_switch** (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _BinaryFunction2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism=__gnu_parallel::parallel_unbal
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation, typename _IteratorTag1, typename _IteratorTag2>
  _OutputIterator **std::__parallel::__partial_sum_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::__partial_sum_switch** (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOperation>
  _Tp **std::__parallel::accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op)
- template<typename _IIter, typename _Tp, typename _BinaryOperation>
  _Tp **std::__parallel::accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOperation>
  _Tp **std::__parallel::accumulate** (_IIter __begin, _IIter __end, _Tp __init, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp)
- template<typename _IIter, typename _Tp>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _Tp>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)

- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _↩
  BinaryOperation __bin_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _↩
  BinaryOperation __binary_op)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::adjacent_difference** (_IIter __begin, _IIter __end, _OutputIterator __result, _↩
  BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, _Binary↩
  Operation __bin_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::partial_sum** (_IIter __begin, _IIter __end, _OutputIterator __result, _Binary↩
  Operation __binary_op)

7.616.1 Detailed Description

Parallel STL function calls corresponding to stl_numeric.h. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

7.617 experimental/optional File Reference

Classes

- class std::experimental::fundamentals_v1::bad_optional_access
- struct std::hash< experimental::optional< _Tp > >
- struct std::experimental::fundamentals_v1::in_place_t
- struct std::experimental::fundamentals_v1::nullopt_t
- class std::experimental::fundamentals_v1::optional< _Tp >

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_optional**
- #define **_GLIBCXX_EXPERIMENTAL_OPTIONAL**

Variables

- constexpr in_place_t std::experimental::in_place
- constexpr nullopt_t std::experimental::nullopt

7.617.1 Detailed Description

This is a TS C++ Library header.

7.618 optional File Reference

7.618.1 Detailed Description

This is a Standard C++ Library header.

7.619 ostream File Reference

Namespaces

- namespace std

Functions

- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::endl (basic_ostream< _CharT, _Traits > &__os)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::ends (basic_ostream< _CharT, _Traits > &__os)
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & std::flush (basic_ostream< _CharT, _Traits > &__os)

7.619.1 Detailed Description

This is a Standard C++ Library header.

7.620 algo.h File Reference

Classes

- struct std::__parallel::_CRandNumber< _MustBeInt >

Namespaces

- namespace std
- namespace std::__parallel

Functions

- template<typename _FIterator, typename _BinaryPredicate, typename _IteratorTag>
  _FIterator **std::__parallel::__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _BinaryPredicate ↩
  __pred, _IteratorTag)

- template<typename _FIterator, typename _IteratorTag>
  _FIterator **std::__parallel::__adjacent_find_switch** (_FIterator __begin, _FIterator __end, _IteratorTag)
- template<typename _RAIter, typename _BinaryPredicate>
  _RAIter **std::__parallel::__adjacent_find_switch** (_RAIter __begin, _RAIter __end, _BinaryPredicate __pred, random_access_iterator_tag)
- template<typename _RAIter>
  _RAIter **std::__parallel::__adjacent_find_switch** (_RAIter __begin, _RAIter __end, random_access_iterator_tag)
- template<typename _IIter, typename _Predicate, typename _IteratorTag>
  iterator_traits< _IIter >::difference_type **std::__parallel::__count_if_switch** (_IIter __begin, _IIter __end, _↩
  Predicate __pred, _IteratorTag)
- template<typename _RAIter, typename _Predicate>
  iterator_traits< _RAIter >::difference_type **std::__parallel::__count_if_switch** (_RAIter __begin, _RAIter __↩
  end, _Predicate __pred, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp, typename _IteratorTag>
  iterator_traits< _IIter >::difference_type **std::__parallel::__count_switch** (_IIter __begin, _IIter __end, const _Tp &__value, _IteratorTag)
- template<typename _RAIter, typename _Tp>
  iterator_traits< _RAIter >::difference_type **std::__parallel::__count_switch** (_RAIter __begin, _RAIter __end, const _Tp &__value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>
  _IIter **std::__parallel::__find_first_of_switch** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _BinaryPredicate __comp, _IteratorTag1, _IteratorTag2)
- template<typename _IIter, typename _FIterator, typename _IteratorTag1, typename _IteratorTag2>
  _IIter **std::__parallel::__find_first_of_switch** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter, typename _FIterator, typename _BinaryPredicate, typename _IteratorTag>
  _RAIter **std::__parallel::__find_first_of_switch** (_RAIter __begin1, _RAIter __end1, _FIterator __begin2, _↩
  FIterator __end2, _BinaryPredicate __comp, random_access_iterator_tag, _IteratorTag)
- template<typename _IIter, typename _Predicate, typename _IteratorTag>
  _IIter **std::__parallel::__find_if_switch** (_IIter __begin, _IIter __end, _Predicate __pred, _IteratorTag)
- template<typename _RAIter, typename _Predicate>
  _RAIter **std::__parallel::__find_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _IIter, typename _Tp, typename _IteratorTag>
  _IIter **std::__parallel::__find_switch** (_IIter __begin, _IIter __end, const _Tp &__val, _IteratorTag)
- template<typename _RAIter, typename _Tp>
  _RAIter **std::__parallel::__find_switch** (_RAIter __begin, _RAIter __end, const _Tp &__val, random_access_iterator_tag)
- template<typename _IIter, typename _Function, typename _IteratorTag>
  _Function **std::__parallel::__for_each_switch** (_IIter __begin, _IIter __end, _Function __f, _IteratorTag)
- template<typename _RAIter, typename _Function>
  _Function **std::__parallel::__for_each_switch** (_RAIter __begin, _RAIter __end, _Function __f, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _OutputIterator, typename _Size, typename _Generator, typename _IteratorTag>
  _OutputIterator **std::__parallel::__generate_n_switch** (_OutputIterator __begin, _Size __n, _Generator __gen, _IteratorTag)
- template<typename _RAIter, typename _Size, typename _Generator>
  _RAIter **std::__parallel::__generate_n_switch** (_RAIter __begin, _Size __n, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Generator, typename _IteratorTag>
  void **std::__parallel::__generate_switch** (_FIterator __begin, _FIterator __end, _Generator __gen, _IteratorTag)
- template<typename _RAIter, typename _Generator>
  void **std::__parallel::__generate_switch** (_RAIter __begin, _RAIter __end, _Generator __gen, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)

- template<typename _FIterator, typename _Compare, typename _IteratorTag>
  _FIterator **std::__parallel::__max_element_switch** (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)
- template<typename _RAIter, typename _Compare>
  _RAIter **std::__parallel::__max_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>
  _OutputIterator **std::__parallel::__merge_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Compare __comp, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare>
  _OutputIterator **std::__parallel::__merge_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 _end2, _OutputIterator __result, _Compare __comp, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator, typename _Compare, typename _IteratorTag>
  _FIterator **std::__parallel::__min_element_switch** (_FIterator __begin, _FIterator __end, _Compare __comp, _IteratorTag)
- template<typename _RAIter, typename _Compare>
  _RAIter **std::__parallel::__min_element_switch** (_RAIter __begin, _RAIter __end, _Compare __comp, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Predicate, typename _IteratorTag>
  _FIterator **std::__parallel::__partition_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, _IteratorTag)
- template<typename _RAIter, typename _Predicate>
  _RAIter **std::__parallel::__partition_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, random_access_iterator_tag)
- template<typename _FIterator, typename _Predicate, typename _Tp, typename _IteratorTag>
  void **std::__parallel::__replace_if_switch** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new_value, _IteratorTag)
- template<typename _RAIter, typename _Predicate, typename _Tp>
  void **std::__parallel::__replace_if_switch** (_RAIter __begin, _RAIter __end, _Predicate __pred, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Tp, typename _IteratorTag>
  void **std::__parallel::__replace_switch** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__new_value, _IteratorTag)
- template<typename _RAIter, typename _Tp>
  void **std::__parallel::__replace_switch** (_RAIter __begin, _RAIter __end, const _Tp &__old_value, const _Tp &__new_value, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate, typename _IteratorTag>
  _FIterator **std::__parallel::__search_n_switch** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, _IteratorTag)
- template<typename _RAIter, typename _Integer, typename _Tp, typename _BinaryPredicate>
  _RAIter **std::__parallel::__search_n_switch** (_RAIter __begin, _RAIter __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, random_access_iterator_tag)
- template<typename _FIterator1, typename _FIterator2, typename _IteratorTag1, typename _IteratorTag2>
  _FIterator1 **std::__parallel::__search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2>
  _RAIter1 **std::__parallel::__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>
  _OutputIterator **std::__parallel::__set_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)

- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>
  _OutputIterator **std::__parallel::__set_intersection_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_intersection_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>
  _OutputIterator **std::__parallel::__set_symmetric_difference_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_symmetric_difference_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _OutputIterator, typename _IteratorTag1, typename _IteratorTag2, typename _IteratorTag3>
  _OutputIterator **std::__parallel::__set_union_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __result, _Predicate __pred, _IteratorTag1, _IteratorTag2, _IteratorTag3)
- template<typename _RAIter1, typename _RAIter2, typename _Output_RAIter, typename _Predicate>
  _Output_RAIter **std::__parallel::__set_union_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Output_RAIter __result, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation, typename _IteratorTag1, typename _IteratorTag2>
  _RAIter2 **std::__parallel::__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _UnaryOperation>
  _RAIter2 **std::__parallel::__transform1_switch** (_RAIter1 __begin, _RAIter1 __end, _RAIter2 __result, _UnaryOperation __unary_op, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation, typename _Tag1, typename _Tag2, typename _Tag3>
  _OutputIterator **std::__parallel::__transform2_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, _Tag1, _Tag2, _Tag3)
- template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _BinaryOperation>
  _RAIter3 **std::__parallel::__transform2_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter3 __result, _BinaryOperation __binary_op, random_access_iterator_tag, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _OutputIterator, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  _OutputIterator **std::__parallel::__unique_copy_switch** (_IIter __begin, _IIter __last, _OutputIterator __out, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter, typename _RandomAccessOutputIterator, typename _Predicate>
  _RandomAccessOutputIterator **std::__parallel::__unique_copy_switch** (_RAIter __begin, _RAIter __last, _RandomAccessOutputIterator __out, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator>
  _FIterator **std::__parallel::adjacent_find** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator>
  _FIterator **std::__parallel::adjacent_find** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)

- template<typename _FIterator, typename _BinaryPredicate>
  _FIterator **std::__parallel::adjacent_find** (_FIterator __begin, _FIterator __end, _BinaryPredicate __binary_↩
  pred, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _BinaryPredicate>
  _FIterator **std::__parallel::adjacent_find** (_FIterator __begin, _FIterator __end, _BinaryPredicate __pred)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__↩
  value)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__↩
  value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Tp>
  iterator_traits< _IIter >::difference_type **std::__parallel::count** (_IIter __begin, _IIter __end, const _Tp &__↩
  value, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __↩
  pred)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __↩
  pred, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _Predicate>
  iterator_traits< _IIter >::difference_type **std::__parallel::count_if** (_IIter __begin, _IIter __end, _Predicate __↩
  pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp>
  _IIter **std::__parallel::find** (_IIter __begin, _IIter __end, const _Tp &__val)
- template<typename _IIter, typename _Tp>
  _IIter **std::__parallel::find** (_IIter __begin, _IIter __end, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIterator>
  _IIter **std::__parallel::find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2)
- template<typename _IIter, typename _FIterator>
  _IIter **std::__parallel::find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2,
  __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _FIterator, typename _BinaryPredicate>
  _IIter **std::__parallel::find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _↩
  BinaryPredicate __comp)
- template<typename _IIter, typename _FIterator, typename _BinaryPredicate>
  _IIter **std::__parallel::find_first_of** (_IIter __begin1, _IIter __end1, _FIterator __begin2, _FIterator __end2, _↩
  BinaryPredicate __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Predicate>
  _IIter **std::__parallel::find_if** (_IIter __begin, _IIter __end, _Predicate __pred)
- template<typename _IIter, typename _Predicate>
  _IIter **std::__parallel::find_if** (_IIter __begin, _IIter __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Function>
  _Function **std::__parallel::for_each** (_IIter __begin, _IIter __end, _Function __f, __gnu_parallel::sequential_tag)
- template<typename _Iterator, typename _Function>
  _Function **std::__parallel::for_each** (_Iterator __begin, _Iterator __end, _Function __f)
- template<typename _Iterator, typename _Function>
  _Function **std::__parallel::for_each** (_Iterator __begin, _Iterator __end, _Function __f, __gnu_parallel::_Parallelism
  __parallelism_tag)
- template<typename _FIterator, typename _Generator>
  void **std::__parallel::generate** (_FIterator __begin, _FIterator __end, _Generator __gen)

- template<typename _FIterator, typename _Generator>
  void **std::__parallel::generate** (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Generator>
  void **std::__parallel::generate** (_FIterator __begin, _FIterator __end, _Generator __gen, __gnu_parallel::sequential_tag)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  _OutputIterator **std::__parallel::generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  _OutputIterator **std::__parallel::generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _OutputIterator, typename _Size, typename _Generator>
  _OutputIterator **std::__parallel::generate_n** (_OutputIterator __begin, _Size __n, _Generator __gen, __gnu_parallel::sequential_tag)
- template<typename _FIterator>
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator>
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::_Parallelism ↩ __parallelism_tag)
- template<typename _FIterator>
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, _Compare __comp)
- template<typename _FIterator, typename _Compare>
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **std::__parallel::max_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _↩ OutputIterator __result)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _↩ OutputIterator __result, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare>
  _OutputIterator **std::__parallel::merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _↩ OutputIterator __result, _Compare __comp)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Compare>
  _OutputIterator **std::__parallel::merge** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _↩ OutputIterator __result, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _FIterator>
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end)
- template<typename _FIterator>
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::_Parallelism _↩ _parallelism_tag)
- template<typename _FIterator>
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Compare>
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, _Compare __comp)
- template<typename _FIterator, typename _Compare>
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::_Parallelism __parallelism_tag)

- template<typename _FIterator, typename _Compare>
  _FIterator **std::__parallel::min_element** (_FIterator __begin, _FIterator __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::nth_element** (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::partial_sort** (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Predicate>
  _FIterator **std::__parallel::partition** (_FIterator __begin, _FIterator __end, _Predicate __pred)
- template<typename _FIterator, typename _Predicate>
  _FIterator **std::__parallel::partition** (_FIterator __begin, _FIterator __end, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &&__rand)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void **std::__parallel::random_shuffle** (_RAIter __begin, _RAIter __end, _RandomNumberGenerator &__rand, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Tp>
  void **std::__parallel::replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__↩
  new_value)
- template<typename _FIterator, typename _Tp>
  void **std::__parallel::replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__↩
  new_value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Tp>
  void **std::__parallel::replace** (_FIterator __begin, _FIterator __end, const _Tp &__old_value, const _Tp &__↩
  new_value, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Predicate, typename _Tp>
  void **std::__parallel::replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new↩
  _value)
- template<typename _FIterator, typename _Predicate, typename _Tp>
  void **std::__parallel::replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new↩
  _value, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _FIterator, typename _Predicate, typename _Tp>
  void **std::__parallel::replace_if** (_FIterator __begin, _FIterator __end, _Predicate __pred, const _Tp &__new↩
  _value, __gnu_parallel::sequential_tag)

- template<typename _FIterator1, typename _FIterator2>
  _FIterator1 **std::__parallel::search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2)
- template<typename _FIterator1, typename _FIterator2>
  _FIterator1 **std::__parallel::search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, __gnu_parallel::sequential_tag)
- template<typename _ForwardIterator, typename _Searcher>
  _ForwardIterator std::__parallel::search (_ForwardIterator __first, _ForwardIterator __last, const _Searcher &_↩ _searcher)
- template<typename _FIterator, typename _Integer, typename _Tp>
  _FIterator **std::__parallel::search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val)
- template<typename _FIterator, typename _Integer, typename _Tp>
  _FIterator **std::__parallel::search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, __gnu_parallel::sequential_tag)
- template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
  _FIterator **std::__parallel::search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred)
- template<typename _FIterator, typename _Integer, typename _Tp, typename _BinaryPredicate>
  _FIterator **std::__parallel::search_n** (_FIterator __begin, _FIterator __end, _Integer __count, const _Tp &__val, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __↩ end2, _OutputIterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __↩ end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __↩ end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __↩ end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 ↩ __end2, _OutputIterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 ↩ __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 ↩ __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_intersection** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 ↩ __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)

- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_symmetric_difference** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator>
  _OutputIterator **std::__parallel::set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::set_union** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_exact_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_sampling_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare, typename _Parallelism>
  void **std::__parallel::sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __parallelism)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::balanced_quicksort_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::default_parallel_tag __parallelism)

- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::multiway_mergesort_tag ↩ __parallelism)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::parallel_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::quicksort_tag __parallelism)
- template<typename _RAIter>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIter, typename _Compare, typename _Parallelism>
  void **std::__parallel::stable_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __↩ parallelism)
- template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>
  _OutputIterator **std::__parallel::transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _Unary↩ Operation __unary_op)
- template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>
  _OutputIterator **std::__parallel::transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _Unary↩ Operation __unary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter, typename _OutputIterator, typename _UnaryOperation>
  _OutputIterator **std::__parallel::transform** (_IIter __begin, _IIter __end, _OutputIterator __result, _Unary↩ Operation __unary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::_Parallelism __parallelism_tag)
- template<typename _IIter1, typename _IIter2, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator **std::__parallel::transform** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _OutputIterator __result, _BinaryOperation __binary_op, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out)
- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **std::__parallel::unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, _Predicate __pred)
- template<typename _IIter, typename _OutputIterator, typename _Predicate>
  _OutputIterator **std::__parallel::unique_copy** (_IIter __begin1, _IIter __end1, _OutputIterator __out, _Predicate __pred, __gnu_parallel::sequential_tag)

7.620.1 Detailed Description

Parallel STL function calls corresponding to the stl_algo.h header.
The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

7.621 algobase.h File Reference

Namespaces

- namespace std
- namespace std::__parallel

Functions

- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  bool **std::__parallel::__equal_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  bool **std::__parallel::__equal_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  bool **std::__parallel::__lexicographical_compare_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  bool **std::__parallel::__lexicographical_compare_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  pair< _IIter1, _IIter2 > **std::__parallel::__mismatch_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _IIter1, typename _IIter2, typename _Predicate, typename _IteratorTag1, typename _IteratorTag2>
  pair< _IIter1, _IIter2 > **std::__parallel::__mismatch_switch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  pair< _RAIter1, _RAIter2 > **std::__parallel::__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _RAIter1, typename _RAIter2, typename _Predicate>
  pair< _RAIter1, _RAIter2 > **std::__parallel::__mismatch_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _Predicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate, typename _IteratorTag1, typename _IteratorTag2>
  _FIterator1 **std::__parallel::__search_switch** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, _IteratorTag1, _IteratorTag2)
- template<typename _RAIter1, typename _RAIter2, typename _BinaryPredicate>
  _RAIter1 **std::__parallel::__search_switch** (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _RAIter2 __end2, _BinaryPredicate __pred, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1, typename _IIter2>
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template<typename _IIter1, typename _IIter2>
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  constexpr bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _BinaryPredicate __binary_pred)
- template<typename _IIter1, typename _IIter2, typename _BinaryPredicate>
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)

- template<typename _IIter1, typename _IIter2, typename _Predicate>
  constexpr bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  bool **std::__parallel::equal** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  constexpr bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template<typename _IIter1, typename _IIter2>
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 _↩ _end2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  constexpr bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2, _Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  bool **std::__parallel::lexicographical_compare** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 _↩ _end2, _Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _IIter2 __end2)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _↩ Predicate __pred)
- template<typename _IIter1, typename _IIter2, typename _Predicate>
  pair< _IIter1, _IIter2 > **std::__parallel::mismatch** (_IIter1 __begin1, _IIter1 __end1, _IIter2 __begin2, _↩ Predicate __pred, __gnu_parallel::sequential_tag)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  pair< _InputIterator1, _InputIterator2 > **std::__parallel::mismatch** (_InputIterator1 __begin1, _InputIterator1 ↩ __end1, _InputIterator2 __begin2, _InputIterator2 __end2, _BinaryPredicate __binary_pred)
- template<typename _InputIterator1, typename _InputIterator2>
  pair< _InputIterator1, _InputIterator2 > **std::__parallel::mismatch** (_InputIterator1 __first1, _InputIterator1 _↩ _last1, _InputIterator2 __first2, _InputIterator2 __last2, __gnu_parallel::sequential_tag)
- template<typename _InputIterator1, typename _InputIterator2, typename _BinaryPredicate>
  pair< _InputIterator1, _InputIterator2 > **std::__parallel::mismatch** (_InputIterator1 __first1, _InputIterator1 __↩ last1, _InputIterator2 __first2, _InputIterator2 __last2, _BinaryPredicate __binary_pred, __gnu_parallel::sequential_tag)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate>
  constexpr _FIterator1 **std::__parallel::search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred)
- template<typename _FIterator1, typename _FIterator2, typename _BinaryPredicate>
  _FIterator1 **std::__parallel::search** (_FIterator1 __begin1, _FIterator1 __end1, _FIterator2 __begin2, _FIterator2 __end2, _BinaryPredicate __pred, __gnu_parallel::sequential_tag)

7.621.1 Detailed Description

Parallel STL function calls corresponding to the stl_algobase.h header. The functions defined here mainly do case switches and call the actual parallelized versions in other files. Inlining policy: Functions that basically only contain one function call, are declared inline. This file is a GNU parallel extension to the Standard C++ Library.

7.622 balanced_quicksort.h File Reference

Classes

- struct __gnu_parallel::_QSBThreadLocal< _RAIter >

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort_qsb (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__qsb_conquer (_QSBThreadLocal< _RAIter > **__tls, _RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __iam, _ThreadIndex __num_threads, bool __parent_wait)
- template<typename _RAIter, typename _Compare>
  std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__qsb_divide (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__qsb_local_sort_with_helping (_QSBThreadLocal< _RAIter > **__tls, _Compare &__comp, _ThreadIndex __iam, bool __wait)

7.622.1 Detailed Description

Implementation of a dynamically load-balanced parallel quicksort.
It works in-place and needs only logarithmic extra memory. The algorithm is similar to the one proposed in
P. Tsigas and Y. Zhang. A simple, fast parallel implementation of quicksort and its performance evaluation on SUN enterprise 10000. In 11th Euromicro Conference on Parallel, Distributed and Network-Based Processing, page 372, 2003.
This file is a GNU parallel extension to the Standard C++ Library.

7.623 base.h File Reference

Classes

- class __gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >
- class __gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >
- class __gnu_parallel::__unary_negate< _Predicate, argument_type >
- class __gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >
- struct __gnu_parallel::_EqualTo< _T1, _T2 >
- struct __gnu_parallel::_Less< _T1, _T2 >
- struct __gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >
- struct __gnu_parallel::_Plus< _Tp1, _Tp2, _Result >
- class __gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >
- class __gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >

Namespaces

- namespace __gnu_parallel
- namespace __gnu_sequential
- namespace std
- namespace std::__parallel

Macros

- #define **_GLIBCXX_PARALLEL_ASSERT**(_Condition)

Functions

- void __gnu_parallel::__decode2 (_CASable __x, int &__a, int &__b)
- _CASable __gnu_parallel::__encode2 (int __a, int __b)
- _ThreadIndex **__gnu_parallel::__get_max_threads** ()
- bool **__gnu_parallel::__is_parallel** (const _Parallelism __p)
- template<typename _RAIter, typename _Compare>
  _RAIter __gnu_parallel::__median_of_three_iterators (_RAIter __a, _RAIter __b, _RAIter __c, _Compare __↩
  comp)
- template<typename _Size>
  _Size __gnu_parallel::__rd_log2 (_Size __n)
- template<typename _Tp>
  const _Tp & __gnu_parallel::max (const _Tp &__a, const _Tp &__b)
- template<typename _Tp>
  const _Tp & __gnu_parallel::min (const _Tp &__a, const _Tp &__b)

7.623.1 Detailed Description

Sequential helper functions. This file is a GNU parallel extension to the Standard C++ Library.

7.624 basic_iterator.h File Reference

7.624.1 Detailed Description

Includes the original header files concerned with iterators except for stream iterators. This file is a GNU parallel extension to the Standard C++ Library.

7.625 checkers.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter, typename _Compare>
  bool __gnu_parallel::__is_sorted (_IIter __begin, _IIter __end, _Compare __comp)

7.625.1 Detailed Description

Routines for checking the correctness of algorithm results. This file is a GNU parallel extension to the Standard C++ Library.

7.626 compiletime_settings.h File Reference

Macros

- #define _GLIBCXX_CALL(__n)
- #define _GLIBCXX_PARALLEL_ASSERTIONS
- #define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1
- #define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB
- #define _GLIBCXX_SCALE_DOWN_FPU
- #define _GLIBCXX_VERBOSE_LEVEL

7.626.1 Detailed Description

Defines on options concerning debugging and performance, at compile-time. This file is a GNU parallel extension to the Standard C++ Library.

7.626.2 Macro Definition Documentation

_GLIBCXX_CALL

```
#define _GLIBCXX_CALL(
                __n)
```

Macro to produce log message when entering a function.

Parameters

_↩ _n	Input size.

See also

_GLIBCXX_VERBOSE_LEVEL

Referenced by __gnu_parallel::__find_template(), __gnu_parallel::__find_template(), __gnu_parallel::__find_template(), __gnu_parallel::__for_each_template_random_access_workstealing(), __gnu_parallel::__merge_advance(), __gnu_parallel::__parallel_n __gnu_parallel::__parallel_partial_sum(), __gnu_parallel::__parallel_partition(), __gnu_parallel::__parallel_random_shuffle_drs(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort(), __gnu_parallel::__parallel_sort_qs() __gnu_parallel::__parallel_sort_qsb(), __gnu_parallel::__parallel_unique_copy(), __gnu_parallel::__search_template(), __gnu_parallel::__sequential_multiway_merge(), __gnu_parallel::multiseq_partition(), __gnu_parallel::multiseq_selection(), __gnu_parallel::multiway_merge(), __gnu_parallel::multiway_merge_3_variant(), __gnu_parallel::multiway_merge_4_variant(), __gnu_parallel::multiway_merge_loser_tree(), __gnu_parallel::multiway_merge_loser_tree_sentinel(), __gnu_parallel::multiway_merge_lo __gnu_parallel::multiway_merge_sentinels(), __gnu_parallel::parallel_multiway_merge(), and __gnu_parallel::parallel_sort_mwms().

_GLIBCXX_PARALLEL_ASSERTIONS

```
#define _GLIBCXX_PARALLEL_ASSERTIONS
```

Switch on many _GLIBCXX_PARALLEL_ASSERTions in parallel code. Should be switched on only locally.
Referenced by __gnu_parallel::__qsb_local_sort_with_helping().

_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1

```
#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1
```

Switch on many _GLIBCXX_PARALLEL_ASSERTions in parallel code. Consider the size of the L1 cache for gnu_↩ parallel::__parallel_random_shuffle().

_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB

```
#define _GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB
```

Switch on many _GLIBCXX_PARALLEL_ASSERTions in parallel code. Consider the size of the TLB for gnu_parallel::↩ __parallel_random_shuffle().

_GLIBCXX_SCALE_DOWN_FPU

```
#define _GLIBCXX_SCALE_DOWN_FPU
```

Use floating-point scaling instead of modulo for mapping random numbers to a range. This can be faster on certain CPUs.

_GLIBCXX_VERBOSE_LEVEL

```
#define _GLIBCXX_VERBOSE_LEVEL
```

Determine verbosity level of the parallel mode. Level 1 prints a message each time a parallel-mode function is entered.

7.627 equally_split.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _DifferenceType, typename _OutputIterator>
  _OutputIterator __gnu_parallel::__equally_split (_DifferenceType __n, _ThreadIndex __num_threads, _Output↩
  Iterator __s)
- template<typename _DifferenceType>
  _DifferenceType __gnu_parallel::__equally_split_point (_DifferenceType __n, _ThreadIndex __num_threads, _ThreadIndex __thread_no)

7.627.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

7.628 features.h File Reference

Macros

- #define _GLIBCXX_BAL_QUICKSORT
- #define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS
- #define _GLIBCXX_FIND_EQUAL_SPLIT
- #define _GLIBCXX_FIND_GROWING_BLOCKS
- #define _GLIBCXX_MERGESORT
- #define _GLIBCXX_QUICKSORT
- #define _GLIBCXX_TREE_DYNAMIC_BALANCING
- #define _GLIBCXX_TREE_FULL_COPY
- #define _GLIBCXX_TREE_INITIAL_SPLITTING

7.628.1 Detailed Description

Defines on whether to include algorithm variants.
Less variants reduce executable size and compile time. This file is a GNU parallel extension to the Standard C++ Library.

7.628.2 Macro Definition Documentation

_GLIBCXX_BAL_QUICKSORT

```
#define _GLIBCXX_BAL_QUICKSORT
```

Include parallel dynamically load-balanced quicksort.

See also

__gnu_parallel::_Settings::sort_algorithm

_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS

```
#define _GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS
```

Include the equal-sized blocks variant for std::find.

See also

__gnu_parallel::_Settings::find_algorithm

_GLIBCXX_FIND_EQUAL_SPLIT

```
#define _GLIBCXX_FIND_EQUAL_SPLIT
```

Include the equal splitting variant for std::find.

See also

__gnu_parallel::_Settings::find_algorithm

_GLIBCXX_FIND_GROWING_BLOCKS

```
#define _GLIBCXX_FIND_GROWING_BLOCKS
```

Include the growing blocks variant for std::find.

See also

__gnu_parallel::_Settings::find_algorithm

_GLIBCXX_MERGESORT

```
#define _GLIBCXX_MERGESORT
```

Include parallel multi-way mergesort.

See also

__gnu_parallel::_Settings::sort_algorithm

_GLIBCXX_QUICKSORT

```
#define _GLIBCXX_QUICKSORT
```

Include parallel unbalanced quicksort.

See also

__gnu_parallel::_Settings::sort_algorithm

_GLIBCXX_TREE_DYNAMIC_BALANCING

```
#define _GLIBCXX_TREE_DYNAMIC_BALANCING
```

Include the dynamic balancing variant for _Rb_tree::insert_unique(_IIter beg, _IIter __end).

See also

__gnu_parallel::_Rb_tree

_GLIBCXX_TREE_FULL_COPY

```
#define _GLIBCXX_TREE_FULL_COPY
```

In order to sort the input sequence of _Rb_tree::insert_unique(_IIter beg, _IIter __end) a full copy of the input elements is done.

See also

__gnu_parallel::_Rb_tree

_GLIBCXX_TREE_INITIAL_SPLITTING

```
#define _GLIBCXX_TREE_INITIAL_SPLITTING
```

Include the initial splitting variant for _Rb_tree::insert_unique(_IIter beg, _IIter __end).

See also

__gnu_parallel::_Rb_tree

7.629 find.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector> std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector> std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, constant_size_blocks_tag)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector> std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, equal_split_tag)
- template<typename _RAIter1, typename _RAIter2, typename _Pred, typename _Selector> std::pair< _RAIter1, _RAIter2 > __gnu_parallel::__find_template (_RAIter1 __begin1, _RAIter1 __end1, _RAIter2 __begin2, _Pred __pred, _Selector __selector, growing_blocks_tag)

7.629.1 Detailed Description

Parallel implementation base for std::find(), std::equal() and related functions. This file is a GNU parallel extension to the Standard C++ Library.

7.630 find_selectors.h File Reference

Classes

- struct __gnu_parallel::__adjacent_find_selector
- struct __gnu_parallel::__find_first_of_selector< _FIterator >
- struct __gnu_parallel::__find_if_selector
- struct __gnu_parallel::__generic_find_selector
- struct __gnu_parallel::__mismatch_selector

Namespaces

- namespace __gnu_parallel

7.630.1 Detailed Description

_Function objects representing different tasks to be plugged into the parallel find algorithm. This file is a GNU parallel extension to the Standard C++ Library.

7.631 for_each.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter, typename _UserOp, typename _Functionality, typename _Red, typename _Result>
  _UserOp __gnu_parallel::__for_each_template_random_access (_IIter __begin, _IIter __end, _UserOp __user↩
  _op, _Functionality &__functionality, _Red __reduction, _Result __reduction_start, _Result &__output, typename
  std::iterator_traits< _IIter >::difference_type __bound, _Parallelism __parallelism_tag)

7.631.1 Detailed Description

Main interface for embarrassingly parallel functions.
The explicit implementation are in other header files, like workstealing.h, par_loop.h, omp_loop.h, and omp_loop_↩
static.h. This file is a GNU parallel extension to the Standard C++ Library.

7.632 for_each_selectors.h File Reference

Classes

- struct __gnu_parallel::__accumulate_binop_reduct< _BinOp >
- struct __gnu_parallel::__accumulate_selector< _It >
- struct __gnu_parallel::__adjacent_difference_selector< _It >
- struct __gnu_parallel::__count_if_selector< _It, _Diff >
- struct __gnu_parallel::__count_selector< _It, _Diff >
- struct __gnu_parallel::__fill_selector< _It >
- struct __gnu_parallel::__for_each_selector< _It >
- struct __gnu_parallel::__generate_selector< _It >
- struct __gnu_parallel::__generic_for_each_selector< _It >
- struct __gnu_parallel::__identity_selector< _It >
- struct __gnu_parallel::__inner_product_selector< _It, _It2, _Tp >
- struct __gnu_parallel::__max_element_reduct< _Compare, _It >
- struct __gnu_parallel::__min_element_reduct< _Compare, _It >
- struct __gnu_parallel::__replace_if_selector< _It, _Op, _Tp >
- struct __gnu_parallel::__replace_selector< _It, _Tp >
- struct __gnu_parallel::__transform1_selector< _It >
- struct __gnu_parallel::__transform2_selector< _It >
- struct __gnu_parallel::_DummyReduct
- struct __gnu_parallel::_Nothing

Namespaces

- namespace __gnu_parallel

7.632.1 Detailed Description

Functors representing different tasks to be plugged into the generic parallelization methods for embarrassingly parallel functions. This file is a GNU parallel extension to the Standard C++ Library.

7.633 iterator.h File Reference

Classes

- class __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >
- class __gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >

Namespaces

- namespace __gnu_parallel

7.633.1 Detailed Description

Helper iterator classes for the std::transform() functions. This file is a GNU parallel extension to the Standard C++ Library.

7.634 list_partition.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter>
  void __gnu_parallel::__shrink (std::vector< _IIter > &__os_starts, size_t &__count_to_two, size_t &__range_↩ length)
- template<typename _IIter>
  void __gnu_parallel::__shrink_and_double (std::vector< _IIter > &__os_starts, size_t &__count_to_two, size_t &__range_length, const bool __make_twice)
- template<typename _IIter, typename _FunctorType>
  size_t __gnu_parallel::list_partition (const _IIter __begin, const _IIter __end, _IIter *__starts, size_t *__lengths, const int __num_parts, _FunctorType &__f, int __oversampling=0)

7.634.1 Detailed Description

_Functionality to split __sequence referenced by only input iterators. This file is a GNU parallel extension to the Standard C++ Library.

7.635 losertree.h File Reference

Classes

- struct __gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser
- struct __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser
- class __gnu_parallel::_LoserTree< __stable, _Tp, _Compare >
- class __gnu_parallel::_LoserTree< false, _Tp, _Compare >
- class __gnu_parallel::_LoserTreeBase< _Tp, _Compare >
- class __gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >
- class __gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >
- class __gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >
- class __gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >
- class __gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >
- class __gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >
- class __gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >
- class __gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >
- class __gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >

Namespaces

- namespace __gnu_parallel

7.635.1 Detailed Description

Many generic loser tree variants. This file is a GNU parallel extension to the Standard C++ Library.

7.636 merge.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare> _OutputIterator __gnu_parallel::__merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare> _OutputIterator __gnu_parallel::__merge_advance_movc (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &↩ __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare> _OutputIterator __gnu_parallel::__merge_advance_usual (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &↩ __begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter3, typename _Compare> _RAIter3 __gnu_parallel::__parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter1 &__↩ begin2, _RAIter1 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __↩ max_length, _Compare __comp)
- template<typename _RAIter1, typename _RAIter2, typename _RAIter3, typename _Compare> _RAIter3 __gnu_parallel::__parallel_merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__↩ begin2, _RAIter2 __end2, _RAIter3 __target, typename std::iterator_traits< _RAIter1 >::difference_type __↩ max_length, _Compare __comp)

7.636.1 Detailed Description

Parallel implementation of std::merge(). This file is a GNU parallel extension to the Standard C++ Library.

7.637 multiseq_selection.h File Reference

Classes

- class __gnu_parallel::_Lexicographic< _T1, _T2, _Compare >
- class __gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >

Namespaces

- namespace __gnu_parallel

Macros

- #define __**S**(__i)
- #define __**S**(__i)

Functions

- template<typename _RanSeqs, typename _RankType, typename _RankIterator, typename _Compare>
  void __gnu_parallel::multiseq_partition (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankIterator __begin_offsets, _Compare __comp=std::less< typename std::iterator_traits< typename std::iterator_traits< _RanSeqs >::value_type::first_type >::value_type >())
- template<typename _Tp, typename _RanSeqs, typename _RankType, typename _Compare>
  _Tp __gnu_parallel::multiseq_selection (_RanSeqs __begin_seqs, _RanSeqs __end_seqs, _RankType __rank, _RankType &__offset, _Compare __comp=std::less< _Tp >())

7.637.1 Detailed Description

Functions to find elements of a certain global __rank in multiple sorted sequences. Also serves for splitting such sequence sets.
The algorithm description can be found in
P. J. Varman, S. D. Scheufler, B. R. Iyer, and G. R. Ricard. Merging Multiple Lists on Hierarchical-Memory Multiprocessors. Journal of Parallel and Distributed Computing, 12(2):171-177, 1991.
This file is a GNU parallel extension to the Standard C++ Library.

7.638 multiway_merge.h File Reference

Classes

- struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- struct __gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- struct __gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- struct __gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >
- class __gnu_parallel::_GuardedIterator< _RAIter, _Compare >
- struct __gnu_parallel::_LoserTreeTraits< _Tp >
- struct __gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >
- struct __gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >

Namespaces

- namespace __gnu_parallel

Macros

- #define **_GLIBCXX_PARALLEL_DECISION**(__a, __b, __c, __d)
- #define _GLIBCXX_PARALLEL_LENGTH(__s)
- #define **_GLIBCXX_PARALLEL_MERGE_3_CASE**(__a, __b, __c, __c0, __c1)
- #define **_GLIBCXX_PARALLEL_MERGE_4_CASE**(__a, __b, __c, __d, __c0, __c1, __c2)

Functions

- template<typename _RAIter1, typename _RAIter2, typename _OutputIterator, typename _DifferenceTp, typename _Compare>
  _OutputIterator __gnu_parallel::__merge_advance (_RAIter1 &__begin1, _RAIter1 __end1, _RAIter2 &__begin2, _RAIter2 __end2, _OutputIterator __target, _DifferenceTp __max_length, _Compare __comp)
- template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 __gnu_parallel::__sequential_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sampling_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut __gnu_parallel::multiway_merge (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 __gnu_parallel::multiway_merge_3_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)
- template<template< typename _RAI, typename _Cp > class iterator, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 __gnu_parallel::multiway_merge_4_variant (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)
- template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
  void __gnu_parallel::multiway_merge_exact_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces)
- template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 __gnu_parallel::multiway_merge_loser_tree (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _DifferenceTp __length, _Compare __comp)
- template<typename _UnguardedLoserTree, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 __gnu_parallel::multiway_merge_loser_tree_sentinel (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<typename _LT, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Compare>
  _RAIter3 __gnu_parallel::multiway_merge_loser_tree_unguarded (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, const typename std::iterator_traits< typename std::iterator_traits< _RAIterIterator >::value_type::first_type >::value_type &__sentinel, _DifferenceTp __length, _Compare __comp)
- template<bool __stable, typename _RAIterIterator, typename _Compare, typename _DifferenceType>
  void __gnu_parallel::multiway_merge_sampling_splitting (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _DifferenceType __length, _DifferenceType __total_length, _Compare __comp, std::vector< std::pair< _DifferenceType, _DifferenceType > > *__pieces)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)

- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut __gnu_parallel::multiway_merge_sentinels (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<bool __stable, bool __sentinels, typename _RAIterIterator, typename _RAIter3, typename _DifferenceTp, typename _Splitter, typename _Compare>
  _RAIter3 __gnu_parallel::parallel_multiway_merge (_RAIterIterator __seqs_begin, _RAIterIterator __seqs_end, _RAIter3 __target, _Splitter __splitter, _DifferenceTp __length, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::exact_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, __gnu_parallel::sequential_tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, default_parallel_tag __tag)
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, parallel_tag __tag=parallel_tag(0))
- template<typename _RAIterPairIterator, typename _RAIterOut, typename _DifferenceTp, typename _Compare>
  _RAIterOut **__gnu_parallel::stable_multiway_merge_sentinels** (_RAIterPairIterator __seqs_begin, _RAIterPairIterator __seqs_end, _RAIterOut __target, _DifferenceTp __length, _Compare __comp, sampling_tag __tag)

7.638.1 Detailed Description

Implementation of sequential and parallel multiway merge.
Explanations on the high-speed merging routines in the appendix of
P. Sanders. Fast priority queues for cached memory. ACM Journal of Experimental Algorithmics, 5, 2000.
This file is a GNU parallel extension to the Standard C++ Library.

7.638.2 Macro Definition Documentation

_GLIBCXX_PARALLEL_LENGTH

```
#define _GLIBCXX_PARALLEL_LENGTH(
            __s)
```

Length of a sequence described by a pair of iterators.
Referenced by __gnu_parallel::__sequential_multiway_merge(), __gnu_parallel::multiway_merge_exact_splitting(), __gnu_parallel::multiway_merge_loser_tree(), __gnu_parallel::multiway_merge_sampling_splitting(), and __gnu_parallel::parallel_multiwa

7.639 multiway_mergesort.h File Reference

Classes

- struct __gnu_parallel::_Piece< _DifferenceTp >
- struct __gnu_parallel::_PMWMSSortingData< _RAIter >
- struct __gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >
- struct __gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >
- struct __gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _DifferenceTp>
  void __gnu_parallel::__determine_samples (_PMWMSSortingData< _RAIter > *__sd, _DifferenceTp __num↩
  _samples)
- template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
  void __gnu_parallel::parallel_sort_mwms (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex
  __num_threads)
- template<bool __stable, bool __exact, typename _RAIter, typename _Compare>
  void __gnu_parallel::parallel_sort_mwms_pu (_PMWMSSortingData< _RAIter > *__sd, _Compare &__comp)

7.639.1 Detailed Description

Parallel multiway merge sort. This file is a GNU parallel extension to the Standard C++ Library.

7.640 numericfwd.h File Reference

Namespaces

- namespace std
- namespace std::__parallel

Functions

- template<typename _IIter, typename _Tp, typename _BinaryOper, typename _Tag>
  _Tp **std::__parallel::__accumulate_switch** (_IIter, _IIter, _Tp, _BinaryOper, _Tag)
- template<typename _IIter, typename _Tp, typename _Tag>
  _Tp **std::__parallel::__accumulate_switch** (_IIter, _IIter, _Tp, _Tag)
- template<typename _RAIter, typename _Tp, typename _BinaryOper>
  _Tp **std::__parallel::__accumulate_switch** (_RAIter, _RAIter, _Tp, _BinaryOper, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>
  _OIter **std::__parallel::__adjacent_difference_switch** (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::__adjacent_difference_switch** (_IIter, _IIter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism __parallelism=__gnu_parallel::parallel_unbalanced)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2, typename _Tag1, typename _Tag2>
  _Tp **std::__parallel::__inner_product_switch** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _Binary↩ Function2, _Tag1, _Tag2)
- template<typename _RAIter1, typename _RAIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::__inner_product_switch** (_RAIter1, _RAIter1, _RAIter2, _Tp, _BinaryFunction1, _Binary↩ Function2, random_access_iterator_tag, random_access_iterator_tag, __gnu_parallel::_Parallelism=__gnu_parallel::parallel_unbal
- template<typename _IIter, typename _OIter, typename _BinaryOper, typename _Tag1, typename _Tag2>
  _OIter **std::__parallel::__partial_sum_switch** (_IIter, _IIter, _OIter, _BinaryOper, _Tag1, _Tag2)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::__partial_sum_switch** (_IIter, _IIter, _OIter, _BinaryOper, random_access_iterator_tag, random_access_iterator_tag)
- template<typename _IIter, typename _Tp>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp)
- template<typename _IIter, typename _Tp>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _Tp>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _Tp, typename _BinaryOper>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, _BinaryOper)
- template<typename _IIter, typename _Tp, typename _BinaryOper>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _Tp, typename _BinaryOper>
  _Tp **std::__parallel::accumulate** (_IIter, _IIter, _Tp, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::_Parallelism)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::adjacent_difference** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp)

- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _Tp>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, __gnu_parallel::sequential_tag)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::_Parallelism)
- template<typename _IIter1, typename _IIter2, typename _Tp, typename _BinaryFunction1, typename _BinaryFunction2>
  _Tp **std::__parallel::inner_product** (_IIter1, _IIter1, _IIter2, _Tp, _BinaryFunction1, _BinaryFunction2, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::partial_sum** (_IIter, _IIter, _OIter __result)
- template<typename _IIter, typename _OIter>
  _OIter **std::__parallel::partial_sum** (_IIter, _IIter, _OIter, __gnu_parallel::sequential_tag)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::partial_sum** (_IIter, _IIter, _OIter, _BinaryOper)
- template<typename _IIter, typename _OIter, typename _BinaryOper>
  _OIter **std::__parallel::partial_sum** (_IIter, _IIter, _OIter, _BinaryOper, __gnu_parallel::sequential_tag)

7.640.1 Detailed Description

This file is a GNU parallel extension to the Standard C++ Library.

7.641 omp_loop.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __gnu_parallel::__for_each_template_random_access_omp_loop (_RAIter __begin, _RAIter __end, _Op ↩
  __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::↩
  difference_type __bound)

7.641.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop. This file is a GNU parallel extension to the Standard C++ Library.

7.642 omp_loop_static.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __gnu_parallel::__for_each_template_random_access_omp_loop_static (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::↩
  difference_type __bound)

7.642.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of an OpenMP for loop with static scheduling. This file is a GNU parallel extension to the Standard C++ Library.

7.643 par_loop.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __gnu_parallel::__for_each_template_random_access_ed (_RAIter __begin, _RAIter __end, _Op __o, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::difference_type __bound)

7.643.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of equal splitting. This file is a GNU parallel extension to the Standard C++ Library.

7.644 parallel.h File Reference

7.644.1 Detailed Description

End-user include file. Provides advanced settings and tuning options. This file is a GNU parallel extension to the Standard C++ Library.

7.645 partial_sum.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator __gnu_parallel::__parallel_partial_sum (_IIter __begin, _IIter __end, _OutputIterator __result, ↩
  _BinaryOperation __bin_op)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator __gnu_parallel::__parallel_partial_sum_basecase (_IIter __begin, _IIter __end, _OutputIterator __result, _BinaryOperation __bin_op, typename std::iterator_traits< _IIter >::value_type __value)
- template<typename _IIter, typename _OutputIterator, typename _BinaryOperation>
  _OutputIterator __gnu_parallel::__parallel_partial_sum_linear (_IIter __begin, _IIter __end, _OutputIterator __↩
  result, _BinaryOperation __bin_op, typename std::iterator_traits< _IIter >::difference_type __n)

7.645.1 Detailed Description

Parallel implementation of std::partial_sum(), i.e. prefix sums. This file is a GNU parallel extension to the Standard C++ Library.

7.646 partition.h File Reference

Namespaces

- namespace __gnu_parallel

Macros

- #define _GLIBCXX_VOLATILE

Functions

- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_nth_element (_RAIter __begin, _RAIter __nth, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_partial_sort (_RAIter __begin, _RAIter __middle, _RAIter __end, _Compare __comp)
- template<typename _RAIter, typename _Predicate>
  std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_partition (_RAIter __begin, _RAIter __end, _Predicate __pred, _ThreadIndex __num_threads)

7.646.1 Detailed Description

Parallel implementation of std::partition(), std::nth_element(), and std::partial_sort(). This file is a GNU parallel extension to the Standard C++ Library.

7.646.2 Macro Definition Documentation

_GLIBCXX_VOLATILE

```
#define _GLIBCXX_VOLATILE
```

Decide whether to declare certain variables volatile.
Referenced by __gnu_parallel::__parallel_partition().

7.647 queue.h File Reference

Classes

- class __gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >

Namespaces

- namespace __gnu_parallel

Macros

- #define _GLIBCXX_VOLATILE

7.647.1 Detailed Description

Lock-free double-ended queue. This file is a GNU parallel extension to the Standard C++ Library.

7.647.2 Macro Definition Documentation

_GLIBCXX_VOLATILE

```
#define _GLIBCXX_VOLATILE
```

Decide whether to declare certain variable volatile in this file.

7.648 quicksort.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort_qs (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex _↩ _num_threads)
- template<typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort_qs_conquer (_RAIter __begin, _RAIter __end, _Compare __comp, _ThreadIndex __num_threads)
- template<typename _RAIter, typename _Compare>
  std::iterator_traits< _RAIter >::difference_type __gnu_parallel::__parallel_sort_qs_divide (_RAIter __begin, _↩ RAIter __end, _Compare __comp, typename std::iterator_traits< _RAIter >::difference_type __pivot_rank, type- name std::iterator_traits< _RAIter >::difference_type __num_samples, _ThreadIndex __num_threads)

7.648.1 Detailed Description

Implementation of a unbalanced parallel quicksort (in-place). This file is a GNU parallel extension to the Standard C++ Library.

7.649 random_number.h File Reference

Classes

- class __gnu_parallel::_RandomNumber

Namespaces

- namespace __gnu_parallel

7.649.1 Detailed Description

Random number generator based on the Mersenne twister. This file is a GNU parallel extension to the Standard C++ Library.

7.650 random_shuffle.h File Reference

Classes

- struct __gnu_parallel::_DRandomShufflingGlobalData< _RAIter >
- struct __gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >

Namespaces

- namespace __gnu_parallel

Typedefs

- typedef unsigned short __gnu_parallel::_BinIndex

Functions

- template<typename _RAIter, typename _RandomNumberGenerator>
  void __gnu_parallel::__parallel_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumberGenerator __rng=_RandomNumber())
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __gnu_parallel::__parallel_random_shuffle_drs (_RAIter __begin, _RAIter __end, typename std::iterator_traits< _RAIter >::difference_type __n, _ThreadIndex __num_threads, _RandomNumberGenerator &__rng)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __gnu_parallel::__parallel_random_shuffle_drs_pu (_DRSSorterPU< _RAIter, _RandomNumberGenerator > *__pus)
- template<typename _RandomNumberGenerator>
  int __gnu_parallel::__random_number_pow2 (int __logp, _RandomNumberGenerator &__rng)
- template<typename _Tp>
  _Tp __gnu_parallel::__round_up_to_pow2 (_Tp __x)
- template<typename _RAIter, typename _RandomNumberGenerator>
  void __gnu_parallel::__sequential_random_shuffle (_RAIter __begin, _RAIter __end, _RandomNumber↩ Generator &__rng)

7.650.1 Detailed Description

Parallel implementation of std::random_shuffle(). This file is a GNU parallel extension to the Standard C++ Library.

7.651 search.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _RAIter, typename _DifferenceTp>
  void __gnu_parallel::__calc_borders (_RAIter __elements, _DifferenceTp __length, _DifferenceTp *__off)
- template<typename __RAIter1, typename __RAIter2, typename _Pred>
  __RAIter1 __gnu_parallel::__search_template (__RAIter1 __begin1, __RAIter1 __end1, __RAIter2 __begin2, ↩ __RAIter2 __end2, _Pred __pred)

7.651.1 Detailed Description

Parallel implementation base for std::search() and std::search_n(). This file is a GNU parallel extension to the Standard C++ Library.

7.652 set_operations.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter, typename _OutputIterator>
  _OutputIterator **__gnu_parallel::__copy_tail** (std::pair< _IIter, _IIter > __b, std::pair< _IIter, _IIter > __e, _OutputIterator __r)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__gnu_parallel::__parallel_set_difference** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__gnu_parallel::__parallel_set_intersection** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter, typename _OutputIterator, typename _Operation>
  _OutputIterator **__gnu_parallel::__parallel_set_operation** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Operation __op)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__gnu_parallel::__parallel_set_symmetric_difference** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)
- template<typename _IIter, typename _OutputIterator, typename _Compare>
  _OutputIterator **__gnu_parallel::__parallel_set_union** (_IIter __begin1, _IIter __end1, _IIter __begin2, _IIter __end2, _OutputIterator __result, _Compare __comp)

7.652.1 Detailed Description

Parallel implementations of set operations for random-access iterators. This file is a GNU parallel extension to the Standard C++ Library.

7.653 settings.h File Reference

Classes

- struct __gnu_parallel::_Settings

Namespaces

- namespace __gnu_parallel

Macros

- #define _GLIBCXX_PARALLEL_CONDITION(__c)

7.653.1 Detailed Description

Runtime settings and tuning parameters, heuristics to decide whether to use parallelized algorithms.
This file is a GNU parallel extension to the Standard C++ Library.

7.653.2 Deciding whether to run an algorithm in parallel.

There are several ways the user can switch on and off the parallel execution of an algorithm, both at compile- and run-time.
Only sequential execution can be forced at compile-time. This reduces code size and protects code parts that have non-thread-safe side effects.
Ultimately, forcing parallel execution at compile-time makes sense. Often, the sequential algorithm implementation is used as a subroutine, so no reduction in code size can be achieved. Also, the machine the program is run on might have only one processor core, so to avoid overhead, the algorithm is executed sequentially.
To force sequential execution of an algorithm ultimately at compile-time, the user must add the tag gnu_parallel::sequential_tag() to the end of the parameter list, e. g.

```
std::sort(__v.begin(), __v.end(), __gnu_parallel::sequential_tag());
```

This is compatible with all overloaded algorithm variants. No additional code will be instantiated, at all. The same holds for most algorithm calls with iterators not providing random access.
If the algorithm call is not forced to be executed sequentially at compile-time, the decision is made at run-time. The global variable __gnu_parallel::_Settings::algorithm_strategy is checked. It is a tristate variable corresponding to:

- a. force_sequential, meaning the sequential algorithm is executed.
- b. force_parallel, meaning the parallel algorithm is executed.
- c. heuristic

For heuristic, the parallel algorithm implementation is called only if the input size is sufficiently large. For most algorithms, the input size is the (combined) length of the input sequence(__s). The threshold can be set by the user, individually for each algorithm. The according variables are called gnu_parallel::_Settings::[algorithm]_minimal_n .
For some of the algorithms, there are even more tuning options, e. g. the ability to choose from multiple algorithm variants. See below for details.

7.653.3 Macro Definition Documentation

_GLIBCXX_PARALLEL_CONDITION

```
#define _GLIBCXX_PARALLEL_CONDITION(
                __c)
```

Determine at compile(?)-time if the parallel variant of an algorithm should be called.

Parameters

_↩ _c	A condition that is convertible to bool that is overruled by __gnu_parallel::_Settings::algorithm_strategy. Usually a decision based on the input size.

7.654 sort.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<bool __stable, typename _RAIter, typename _Compare, typename _Parallelism>
  void **__gnu_parallel::__parallel_sort** (_RAIter __begin, _RAIter __end, _Compare __comp, _Parallelism __↩ parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, balanced_quicksort_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, default_parallel_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_exact_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_sampling_tag __parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, multiway_mergesort_tag __parallelism)

- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, parallel_tag __↩
  parallelism)
- template<bool __stable, typename _RAIter, typename _Compare>
  void __gnu_parallel::__parallel_sort (_RAIter __begin, _RAIter __end, _Compare __comp, quicksort_tag __↩
  parallelism)

7.654.1 Detailed Description

Parallel sorting algorithm switch. This file is a GNU parallel extension to the Standard C++ Library.

7.655 tags.h File Reference

Classes

- struct __gnu_parallel::balanced_quicksort_tag
- struct __gnu_parallel::balanced_tag
- struct __gnu_parallel::constant_size_blocks_tag
- struct __gnu_parallel::default_parallel_tag
- struct __gnu_parallel::equal_split_tag
- struct __gnu_parallel::exact_tag
- struct __gnu_parallel::find_tag
- struct __gnu_parallel::growing_blocks_tag
- struct __gnu_parallel::multiway_mergesort_exact_tag
- struct __gnu_parallel::multiway_mergesort_sampling_tag
- struct __gnu_parallel::multiway_mergesort_tag
- struct __gnu_parallel::omp_loop_static_tag
- struct __gnu_parallel::omp_loop_tag
- struct __gnu_parallel::parallel_tag
- struct __gnu_parallel::quicksort_tag
- struct __gnu_parallel::sampling_tag
- struct __gnu_parallel::sequential_tag
- struct __gnu_parallel::unbalanced_tag

Namespaces

- namespace __gnu_parallel

7.655.1 Detailed Description

Tags for compile-time selection. This file is a GNU parallel extension to the Standard C++ Library.

7.656 types.h File Reference

Namespaces

- namespace __gnu_parallel

Typedefs

- typedef int64_t __gnu_parallel::_CASable
- typedef uint64_t __gnu_parallel::_SequenceIndex
- typedef uint16_t __gnu_parallel::_ThreadIndex

Enumerations

- enum __gnu_parallel::_AlgorithmStrategy { **heuristic** , **force_sequential** , **force_parallel** }
- enum __gnu_parallel::_FindAlgorithm { **GROWING_BLOCKS** , **CONSTANT_SIZE_BLOCKS** , **EQUAL_SPLIT** }
- enum __gnu_parallel::_MultiwayMergeAlgorithm { **LOSER_TREE** }
- enum __gnu_parallel::_Parallelism { __gnu_parallel::sequential , __gnu_parallel::parallel_unbalanced , __gnu_parallel::parallel_balanced , __gnu_parallel::parallel_omp_loop , __gnu_parallel::parallel_omp_loop_static , __gnu_parallel::parallel_taskqueue }
- enum __gnu_parallel::_PartialSumAlgorithm { **RECURSIVE** , **LINEAR** }
- enum __gnu_parallel::_SortAlgorithm { **MWMS** , **QS** , **QS_BALANCED** }
- enum __gnu_parallel::_SplittingAlgorithm { **SAMPLING** , **EXACT** }

Variables

- static const int __gnu_parallel::_CASable_bits
- static const _CASable __gnu_parallel::_CASable_mask

7.656.1 Detailed Description

Basic types and typedefs. This file is a GNU parallel extension to the Standard C++ Library.

7.657 unique_copy.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _IIter, class _OutputIterator>
  _OutputIterator __gnu_parallel::__parallel_unique_copy (_IIter __first, _IIter __last, _OutputIterator __result)
- template<typename _IIter, class _OutputIterator, class _BinaryPredicate>
  _OutputIterator __gnu_parallel::__parallel_unique_copy (_IIter __first, _IIter __last, _OutputIterator __result, _↩
  BinaryPredicate __binary_pred)

7.657.1 Detailed Description

Parallel implementations of std::unique_copy(). This file is a GNU parallel extension to the Standard C++ Library.

7.658 workstealing.h File Reference

Classes

- struct __gnu_parallel::_Job< _DifferenceTp >

Namespaces

- namespace __gnu_parallel

Macros

- #define **_GLIBCXX_JOB_VOLATILE**

Functions

- template<typename _RAIter, typename _Op, typename _Fu, typename _Red, typename _Result>
  _Op __gnu_parallel::__for_each_template_random_access_workstealing (_RAIter __begin, _RAIter __end, _Op __op, _Fu &__f, _Red __r, _Result __base, _Result &__output, typename std::iterator_traits< _RAIter >::↩ difference_type __bound)

7.658.1 Detailed Description

Parallelization of embarrassingly parallel execution by means of work-stealing.
Work stealing is described in
R. D. Blumofe and C. E. Leiserson. Scheduling multithreaded computations by work stealing. Journal of the ACM, 46(5):720-748, 1999.
This file is a GNU parallel extension to the Standard C++ Library.

7.659 print File Reference

7.659.1 Detailed Description

This is a Standard C++ Library header.

7.660 queue File Reference

7.660.1 Detailed Description

This is a Standard C++ Library header.

7.661 experimental/random File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_randint**
- #define **_GLIBCXX_EXPERIMENTAL_RANDOM**

Functions

- std::default_random_engine & **std::experimental::_S_randint_engine** ()
- template<typename _IntType>
  _IntType **std::experimental::randint** (_IntType __a, _IntType __b)
- void **std::experimental::reseed** ()
- void **std::experimental::reseed** (default_random_engine::result_type __value)

7.661.1 Detailed Description

This is a TS C++ Library header.

7.662 random File Reference

7.662.1 Detailed Description

This is a Standard C++ Library header.

7.663 ranges File Reference

Classes

- class std::ranges::empty_view< _Tp >
- class std::ranges::single_view< _Tp >

Namespaces

- namespace std

Typedefs

- template<typename _Fn, typename... _Iters>
  using **std::ranges::__detail::__func_handle_t**
- template<typename _Wp>
  using **std::ranges::__detail::__iota_diff_t**
- template<bool _Const, typename _Tp>
  using **std::ranges::__detail::__maybe_const_t**
- template<bool _Const, typename _Tp>
  using **std::ranges::__maybe_const_t**
- template<bool _Present, typename _Tp, int _Disc = 0>
  using **std::ranges::__detail::__maybe_present_t**
- template<viewable_range _Range>
  using **std::ranges::views::all_t**
- template<typename _Val>
  using **std::ranges::istream_view**
- template<typename _Range>
  using **std::ranges::keys_view**
- template<typename _Range>
  using **std::ranges::values_view**
- template<typename _Val>
  using **std::ranges::wistream_view**

Functions

- template<typename _Tp>
  constexpr _Tp & **std::ranges::__detail::__as_lvalue** (_Tp &&__t)
- template<typename _Range>
  constexpr auto **std::ranges::views::__detail::__drop_of_repeat_view** (_Range &&, range_difference_t< _Range >)
- template<typename _Tp, typename _Up>
  void **std::ranges::views::__adaptor::__is_range_adaptor_closure_fn** (const _Tp &, const _RangeAdaptorClosure< _Up > &)
- template<typename _Fn, typename... _Iters>
  consteval auto **std::ranges::__detail::__func_handle::__select** ()
- template<typename _Range>
  constexpr auto **std::ranges::views::__detail::__take_of_repeat_view** (_Range &&, range_difference_t< _Range >)
- template<typename _Wp>
  constexpr auto **std::ranges::__detail::__to_signed_like** (_Wp __w) noexcept
- template<typename _Range>
  std::ranges::common_view (_Range &&) -> common_view< views::all_t< _Range > >

- template<typename _Range>
  std::ranges::drop_view (_Range &&, range_difference_t< _Range >) -> drop_view< views::all_t< _Range > >
- template<typename _Range, typename _Pred>
  std::ranges::drop_while_view (_Range &&, _Pred) -> drop_while_view< views::all_t< _Range >, _Pred >
- template<typename _Range, typename _Pred>
  std::ranges::filter_view (_Range &&, _Pred) -> filter_view< views::all_t< _Range >, _Pred >
- template<typename _Winc, typename _Bound>
  std::ranges::iota_view (_Winc, _Bound) -> iota_view< _Winc, _Bound >
- template<typename _Range>
  std::ranges::join_view (_Range &&) -> join_view< views::all_t< _Range > >
- template<typename _Range, typename _Pattern>
  std::ranges::lazy_split_view (_Range &&, _Pattern &&) -> lazy_split_view< views::all_t< _Range >, views::all_t< _Pattern > >
- template<input_range _Range>
  std::ranges::lazy_split_view (_Range &&, range_value_t< _Range >) -> lazy_split_view< views::all_t< _Range >, single_view< range_value_t< _Range > > >
- template<typename _Lhs, typename _Rhs>
  constexpr auto **std::ranges::views::__adaptor::operator|** (_Lhs &&__lhs, _Rhs &&__rhs)
- template<typename _Self, typename _Range>
  constexpr auto **std::ranges::views::__adaptor::operator|** (_Range &&__r, _Self &&__self)
- template<typename _Range>
  std::ranges::ref_view (_Range &) -> ref_view< _Range >
- template<typename _Range>
  std::ranges::reverse_view (_Range &&) -> reverse_view< views::all_t< _Range > >
- template<typename _Tp>
  std::ranges::single_view (_Tp) -> single_view< _Tp >
- template<typename _Range, typename _Pattern>
  std::ranges::split_view (_Range &&, _Pattern &&) -> split_view< views::all_t< _Range >, views::all_t< _Pattern > >
- template<forward_range _Range>
  std::ranges::split_view (_Range &&, range_value_t< _Range >) -> split_view< views::all_t< _Range >, single_view< range_value_t< _Range > > >
- template<typename _Range>
  std::ranges::take_view (_Range &&, range_difference_t< _Range >) -> take_view< views::all_t< _Range > >
- template<typename _Range, typename _Pred>
  std::ranges::take_while_view (_Range &&, _Pred) -> take_while_view< views::all_t< _Range >, _Pred >
- template<typename _Range, typename _Fp>
  std::ranges::transform_view (_Range &&, _Fp) -> transform_view< views::all_t< _Range >, _Fp >

Variables

- template<typename _Range>
  constexpr bool **std::ranges::views::__detail::__is_basic_string_view**
- template<typename _CharT, typename _Traits>
  constexpr bool **std::ranges::views::__detail::__is_basic_string_view**< **basic_string_view**< **_CharT, _Traits** > >
- template<typename _Range>
  constexpr bool **std::ranges::views::__detail::__is_empty_view**
- template<typename _Tp>
  constexpr bool **std::ranges::views::__detail::__is_empty_view**< **empty_view**< **_Tp** > >

- template<typename _Range>
  constexpr bool **std::ranges::views::__detail::__is_iota_view**
- template<typename _Winc, typename _Bound>
  constexpr bool **std::ranges::views::__detail::__is_iota_view**< **iota_view**< **_Winc, _Bound** > >
- template<typename _Range>
  constexpr bool **std::ranges::views::__detail::__is_repeat_view**
- template<typename>
  constexpr bool **std::ranges::views::__detail::__is_reverse_view**
- template<typename _Vp>
  constexpr bool **std::ranges::views::__detail::__is_reverse_view**< **reverse_view**< **_Vp** > >
- template<typename>
  constexpr bool **std::ranges::views::__detail::__is_reversible_subrange**
- template<typename _Iter, subrange_kind _Kind>
  constexpr bool **std::ranges::views::__detail::__is_reversible_subrange**< **subrange**< **reverse_iterator**< ↩
  _Iter >, **reverse_iterator**< **_Iter** >, **_Kind** > >
- template<typename _Range>
  constexpr bool **std::ranges::views::__detail::__is_subrange**
- constexpr _All **std::ranges::views::all**
- constexpr _Common **std::ranges::views::common**
- constexpr _Counted **std::ranges::views::counted**
- constexpr _Drop **std::ranges::views::drop**
- constexpr _DropWhile **std::ranges::views::drop_while**
- template<size_t _Nm>
  constexpr _Elements< _Nm > **std::ranges::views::elements**
- template<typename _Tp>
  constexpr empty_view< _Tp > **std::ranges::views::empty**
- template<typename _Tp>
  constexpr bool **std::ranges::enable_borrowed_range**< **common_view**< **_Tp** > >
- template<typename _Tp>
  constexpr bool **std::ranges::enable_borrowed_range**< **drop_view**< **_Tp** > >
- template<typename _Tp, typename _Pred>
  constexpr bool **std::ranges::enable_borrowed_range**< **drop_while_view**< **_Tp, _Pred** > >
- template<typename _Tp, size_t _Nm>
  constexpr bool **std::ranges::enable_borrowed_range**< **elements_view**< **_Tp, _Nm** > >
- template<typename _Tp>
  constexpr bool **std::ranges::enable_borrowed_range**< **empty_view**< **_Tp** > >
- template<typename _Winc, typename _Bound>
  constexpr bool **std::ranges::enable_borrowed_range**< **iota_view**< **_Winc, _Bound** > >
- template<typename _Tp>
  constexpr bool **std::ranges::enable_borrowed_range**< **owning_view**< **_Tp** > >
- template<typename _Tp>
  constexpr bool **std::ranges::enable_borrowed_range**< **ref_view**< **_Tp** > >
- template<typename _Tp>
  constexpr bool **std::ranges::enable_borrowed_range**< **reverse_view**< **_Tp** > >
- template<typename _Tp>
  constexpr bool **std::ranges::enable_borrowed_range**< **take_view**< **_Tp** > >
- constexpr _Filter **std::ranges::views::filter**
- constexpr _Iota **std::ranges::views::iota**
- template<typename _Tp>
  constexpr _Istream< _Tp > **std::ranges::views::istream**
- constexpr _Join **std::ranges::views::join**
- constexpr auto **std::ranges::views::keys**

- constexpr _LazySplit **std::ranges::views::lazy_split**
- constexpr _Reverse **std::ranges::views::reverse**
- constexpr _Single **std::ranges::views::single**
- constexpr _Split **std::ranges::views::split**
- constexpr _Take **std::ranges::views::take**
- constexpr _TakeWhile **std::ranges::views::take_while**
- constexpr _Transform **std::ranges::views::transform**
- constexpr auto **std::ranges::views::values**

7.663.1 Detailed Description

This is a Standard C++ Library header.

7.664 experimental/ratio File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_RATIO**

Variables

- template<typename _R1, typename _R2>
  constexpr bool **std::experimental::ratio_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::experimental::ratio_greater_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::experimental::ratio_greater_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::experimental::ratio_less_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::experimental::ratio_less_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::experimental::ratio_not_equal_v**

7.664.1 Detailed Description

This is a TS C++ Library header.

7.665 ratio File Reference

Classes

- struct std::ratio< _Num, _Den >
- struct std::ratio_equal< _R1, _R2 >
- struct std::ratio_greater< _R1, _R2 >
- struct std::ratio_greater_equal< _R1, _R2 >
- struct std::ratio_less< _R1, _R2 >
- struct std::ratio_less_equal< _R1, _R2 >
- struct std::ratio_not_equal< _R1, _R2 >

Namespaces

- namespace std

Typedefs

- using **std::atto**
- using **std::centi**
- using **std::deca**
- using **std::deci**
- using **std::exa**
- using **std::femto**
- using **std::giga**
- using **std::hecto**
- using **std::kilo**
- using **std::mega**
- using **std::micro**
- using **std::milli**
- using **std::nano**
- using **std::peta**
- using **std::pico**
- template<typename _R1, typename _R2>
  using std::ratio_add
- template<typename _R1, typename _R2>
  using std::ratio_divide
- template<typename _R1, typename _R2>
  using std::ratio_multiply
- template<typename _R1, typename _R2>
  using std::ratio_subtract
- using **std::tera**

Variables

- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_greater_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_greater_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_less_equal_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_less_v**
- template<typename _R1, typename _R2>
  constexpr bool **std::ratio_not_equal_v**

7.665.1 Detailed Description

This is a Standard C++ Library header.

7.666 tr2/ratio File Reference

Namespaces

- namespace std
- namespace std::tr2

7.666.1 Detailed Description

This is a TR2 C++ Library header.

7.667 experimental/regex File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_REGEX**

Typedefs

- typedef match_results< const char * > **std::experimental::fundamentals_v2::pmr::cmatch**
- template<typename _BidirectionalIterator>
  using **std::experimental::fundamentals_v2::pmr::match_results**
- typedef match_results< string::const_iterator > **std::experimental::fundamentals_v2::pmr::smatch**
- typedef match_results< const wchar_t * > **std::experimental::fundamentals_v2::pmr::wcmatch**
- typedef match_results< wstring::const_iterator > **std::experimental::fundamentals_v2::pmr::wsmatch**

7.667.1 Detailed Description

This is a TS C++ Library header.

7.668 regex File Reference

Namespaces

- namespace std

Typedefs

- using **std::pmr::cmatch**
- template<typename _BidirectionalIterator>
  using **std::pmr::match_results**
- using **std::pmr::smatch**
- using **std::pmr::wcmatch**
- using **std::pmr::wsmatch**

7.668.1 Detailed Description

This is a Standard C++ Library header.

7.669 scoped_allocator File Reference

Classes

- class std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >

Namespaces

- namespace std

7.669.1 Detailed Description

This is a Standard C++ Library header.

7.670 semaphore File Reference

7.670.1 Detailed Description

This is a Standard C++ Library header.

7.671 debug/set File Reference

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_SET**

7.671.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.672 experimental/set File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_SET**

Typedefs

- template<typename _Key, typename _Compare = less<_Key>>
  using **std::experimental::fundamentals_v2::pmr::multiset**
- template<typename _Key, typename _Compare = less<_Key>>
  using **std::experimental::fundamentals_v2::pmr::set**

Functions

- template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (multiset< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Compare, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (set< _Key, _Compare, _Alloc > &__cont, _Predicate __pred)

7.672.1 Detailed Description

This is a TS C++ Library header.

7.673 set File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Key, typename _Cmp = std::less<_Key>>
  using **std::pmr::multiset**
- template<typename _Key, typename _Cmp = std::less<_Key>>
  using **std::pmr::set**

7.673.1 Detailed Description

This is a Standard C++ Library header.

7.674 shared_mutex File Reference

Classes

- class std::shared_lock< _Mutex >
- class std::shared_timed_mutex

Namespaces

- namespace std

7.674.1 Detailed Description

This is a Standard C++ Library header.

7.675 source_location File Reference

7.675.1 Detailed Description

This is a Standard C++ Library header.

7.676 span File Reference

7.676.1 Detailed Description

This is a Standard C++ Library header.

7.677 sstream File Reference

Classes

- class std::basic_istringstream< _CharT, _Traits, _Alloc >
- class std::basic_ostringstream< _CharT, _Traits, _Alloc >
- class std::basic_stringbuf< _CharT, _Traits, _Alloc >
- class std::basic_stringstream< _CharT, _Traits, _Alloc >

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_LVAL_REF_QUAL**
- #define **_GLIBCXX_SSTREAM_ALWAYS_INLINE**

Functions

- template<class _CharT, class _Traits, class _Allocator>
  void std::swap (basic_istringstream< _CharT, _Traits, _Allocator > &__x, basic_istringstream< _CharT, _Traits, _Allocator > &__y)
- template<class _CharT, class _Traits, class _Allocator>
  void std::swap (basic_ostringstream< _CharT, _Traits, _Allocator > &__x, basic_ostringstream< _CharT, _Traits, _Allocator > &__y)
- template<class _CharT, class _Traits, class _Allocator>
  void std::swap (basic_stringbuf< _CharT, _Traits, _Allocator > &__x, basic_stringbuf< _CharT, _Traits, _Allocator > &__y) noexcept(noexcept(__x.swap(__y)))
- template<class _CharT, class _Traits, class _Allocator>
  void std::swap (basic_stringstream< _CharT, _Traits, _Allocator > &__x, basic_stringstream< _CharT, _Traits, _Allocator > &__y)

7.677.1 Detailed Description

This is a Standard C++ Library header.

7.678 stack File Reference

7.678.1 Detailed Description

This is a Standard C++ Library header.

7.679 stdatomic.h File Reference

7.679.1 Detailed Description

This is a Standard C++ Library header.

7.680 stdbit.h File Reference

7.680.1 Detailed Description

This is a Standard C++ Library header.

7.681 stdckdint.h File Reference

7.681.1 Detailed Description

This is a Standard C++ Library header.

7.682 stdexcept File Reference

Classes

- class std::range_error

Namespaces

- namespace std

7.682.1 Detailed Description

This is a Standard C++ Library header.

7.683 stdlib.h File Reference

Functions

- template<typename _Tp>
  _Tp abs (const complex< _Tp > &)
- ldiv_t **div** (long __i, long __j) noexcept

7.683.1 Detailed Description

This is a Standard C++ Library header.

7.683.2 Function Documentation

abs()

```
template<typename _Tp>
_Tp std::abs (
            const complex< _Tp > & __z)  [inline]
```

Return magnitude of *z*.

7.684 stop_token File Reference

7.684.1 Detailed Description

This is a Standard C++ Library header.

7.685 streambuf File Reference

Classes

- class std::basic_streambuf< _CharT, _Traits >

Namespaces

- namespace std

Macros

- #define **_IsUnused**

Functions

- template<typename _CharT, typename _Traits>
  streamsize **std::__copy_streambufs_eof** (basic_streambuf< _CharT, _Traits > *__sbin, basic_streambuf< ↩
  _CharT, _Traits > *__sbout, bool &__ineof)
- template<> streamsize **std::__copy_streambufs_eof** (basic_streambuf< char > *__sbin, basic_streambuf< char > *__sbout, bool &__ineof)
- template<> streamsize **std::__copy_streambufs_eof** (basic_streambuf< wchar_t > *__sbin, basic_streambuf< wchar_t > *__sbout, bool &__ineof)

7.685.1 Detailed Description

This is a Standard C++ Library header.

7.686 debug/string File Reference

Classes

- class __gnu_debug::basic_string< _CharT, _Traits, _Allocator >
- struct std::hash< __gnu_debug::basic_string< _CharT > >

Namespaces

- namespace __gnu_debug
- namespace std

Macros

- #define **__glibcxx_check_string**(_String)
- #define **__glibcxx_check_string_constructor**(_Str)
- #define **__glibcxx_check_string_len**(_String, _Len)
- #define **__glibcxx_check_string_n_constructor**(_Str, _Size)
- #define **_GLIBCXX_DEBUG_STRING**
- #define **_GLIBCXX_DEBUG_VERIFY_STR_COND_AT**(_Cond, _File, _Line, _Func)
- #define **_GLIBCXX_INSERT_RETURNS_ITERATOR**
- #define **_GLIBCXX_INSERT_RETURNS_ITERATOR_ONLY**(expr)

Typedefs

- typedef basic_string< char > **__gnu_debug::string**
- typedef basic_string< char16_t > __gnu_debug::u16string
- typedef basic_string< char32_t > __gnu_debug::u32string
- typedef basic_string< wchar_t > **__gnu_debug::wstring**

Functions

- template<typename _CharT, typename _Integer>
  const _CharT * __gnu_debug::__check_string (const _CharT *__s, _Integer __n, const char *__file, unsigned int __line, const char *__function)

- template<typename _CharT>
  const _CharT * __gnu_debug::__check_string (const _CharT *__s, const char *__file, unsigned int __line, const char *__function)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_istream< _CharT, _Traits > & **__gnu_debug::getline** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_istream< _CharT, _Traits > & **__gnu_debug::getline** (std::basic_istream< _CharT, _Traits > &__is, basic_string< _CharT, _Traits, _Allocator > &__str, _CharT __delim)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator!=** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator!=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **__gnu_debug::operator+** (_CharT __lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **__gnu_debug::operator+** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **__gnu_debug::operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, _CharT __rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **__gnu_debug::operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  basic_string< _CharT, _Traits, _Allocator > **__gnu_debug::operator+** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator<** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator<** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator<** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_ostream< _CharT, _Traits > & **__gnu_debug::operator<<** (std::basic_ostream< _CharT, _Traits > &__os, const basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator<=** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator<=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator<=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string< _CharT, _Traits, _Allocator > &__rhs)

- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator==** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &_↩
  _rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator==** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *_↩
  _rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator==** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string<
  _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator>** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &__↩
  rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator>** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *__↩
  rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator>** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string<
  _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator>=** (const _CharT *__lhs, const basic_string< _CharT, _Traits, _Allocator > &_↩
  _rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator>=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const _CharT *_↩
  _rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  bool **__gnu_debug::operator>=** (const basic_string< _CharT, _Traits, _Allocator > &__lhs, const basic_string<
  _CharT, _Traits, _Allocator > &__rhs)
- template<typename _CharT, typename _Traits, typename _Allocator>
  std::basic_istream< _CharT, _Traits > & **__gnu_debug::operator>>** (std::basic_istream< _CharT, _Traits >
  &__is, basic_string< _CharT, _Traits, _Allocator > &__str)
- template<typename _CharT, typename _Traits, typename _Allocator>
  void **__gnu_debug::swap** (basic_string< _CharT, _Traits, _Allocator > &__lhs, basic_string< _CharT, _Traits,
  _Allocator > &__rhs)

7.686.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.687 experimental/string File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_STRING**

Typedefs

- template<typename _CharT, typename _Traits = char_traits<_CharT>>
  using **std::experimental::fundamentals_v2::pmr::basic_string**
- typedef basic_string< char > **std::experimental::fundamentals_v2::pmr::string**

- typedef basic_string< char16_t > **std::experimental::fundamentals_v2::pmr::u16string**
- typedef basic_string< char32_t > **std::experimental::fundamentals_v2::pmr::u32string**
- typedef basic_string< wchar_t > **std::experimental::fundamentals_v2::pmr::wstring**

Functions

- template<typename _CharT, typename _Traits, typename _Alloc, typename _Up>
  void **std::experimental::erase** (basic_string< _CharT, _Traits, _Alloc > &__cont, const _Up &__value)
- template<typename _CharT, typename _Traits, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (basic_string< _CharT, _Traits, _Alloc > &__cont, _Predicate __pred)

7.687.1 Detailed Description

This is a TS C++ Library header.

7.688 string File Reference

Namespaces

- namespace std

Macros

- #define **__glibcxx_exc_in_string**
- #define **_GLIBCXX_STRING**

Typedefs

- template<typename _CharT, typename _Traits = char_traits<_CharT>>
  using **std::pmr::basic_string**
- using **std::pmr::string**
- using **std::pmr::u16string**
- using **std::pmr::u32string**
- using **std::pmr::wstring**

7.688.1 Detailed Description

This is a Standard C++ Library header.

7.689 experimental/string_view File Reference

Classes

- class std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_string_view**
- #define **_GLIBCXX_EXPERIMENTAL_STRING_VIEW**

Typedefs

- using **std::experimental::string_view**
- using **std::experimental::u16string_view**
- using **std::experimental::u32string_view**
- using **std::experimental::wstring_view**

Functions

- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator!=** (__type_identity_t< basic_string_view< _CharT, _Traits > > ↩
  __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator!=** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator!=** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- constexpr basic_string_view< char > **std::experimental::literals::operator""sv** (const char *__str, size_t __↩
  len) noexcept
- constexpr basic_string_view< char16_t > **std::experimental::literals::operator""sv** (const char16_t *__str, size_t __len) noexcept
- constexpr basic_string_view< char32_t > **std::experimental::literals::operator""sv** (const char32_t *__str, size_t __len) noexcept
- constexpr basic_string_view< wchar_t > **std::experimental::literals::operator""sv** (const wchar_t *__str, size_t __len) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator<** (__type_identity_t< basic_string_view< _CharT, _Traits > > _↩
  _x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator<** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator<** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::experimental::operator<<** (basic_ostream< _CharT, _Traits > &↩
  __os, basic_string_view< _CharT, _Traits > __str)
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator<=** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator<=** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator<=** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator==** (__type_identity_t< basic_string_view< _CharT, _Traits > > ↩
  __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator==** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept

- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator==** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator>** (__type_identity_t< basic_string_view< _CharT, _Traits > > _↩ _x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator>** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator>** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator>=** (__type_identity_t< basic_string_view< _CharT, _Traits > > __x, basic_string_view< _CharT, _Traits > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator>=** (basic_string_view< _CharT, _Traits > __x, __type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept
- template<typename _CharT, typename _Traits>
  constexpr bool **std::experimental::operator>=** (basic_string_view< _CharT, _Traits > __x, basic_string_view< _CharT, _Traits > __y) noexcept

Variables

- template<typename _CharT, typename _Traits>
  constexpr bool **std::ranges::enable_borrowed_range**< **experimental::basic_string_view**< **_CharT, _Traits** > >
- template<typename _CharT, typename _Traits>
  constexpr bool **std::ranges::enable_view**< **experimental::basic_string_view**< **_CharT, _Traits** > >

7.689.1 Detailed Description

This is a TS C++ Library header.

7.690 string_view File Reference

Classes

- class std::basic_string_view< _CharT, _Traits >

Namespaces

- namespace std
- namespace std::literals

Macros

- #define **__glibcxx_exc_in_string**

Typedefs

- using **std::string_view**
- using **std::u16string_view**
- using **std::u32string_view**
- using **std::wstring_view**

Functions

- constexpr size_t **std::__sv_check** (size_t __size, size_t __pos, const char *__s)
- constexpr size_t **std::__sv_limit** (size_t __size, size_t __pos, size_t __off) noexcept
- template<contiguous_iterator _It, sized_sentinel_for< _It > _End>
  std::basic_string_view (_It, _End) -> basic_string_view< iter_value_t< _It > >
- template<ranges::contiguous_range _Range>
  std::basic_string_view (_Range &&) -> basic_string_view< ranges::range_value_t< _Range > >
- constexpr basic_string_view< char > **std::literals::operator""sv** (const char *__str, size_t __len) noexcept
- constexpr basic_string_view< char16_t > **std::literals::operator""sv** (const char16_t *__str, size_t __len) noexcept
- constexpr basic_string_view< char32_t > **std::literals::operator""sv** (const char32_t *__str, size_t __len) noexcept
- constexpr basic_string_view< wchar_t > **std::literals::operator""sv** (const wchar_t *__str, size_t __len) noexcept
- template<typename _CharT, typename _Traits>
  basic_ostream< _CharT, _Traits > & **std::operator<<** (basic_ostream< _CharT, _Traits > &__os, basic_string_view< _CharT, _Traits > __str)
- template<typename _CharT, typename _Traits>
  constexpr auto **std::operator<=>** (basic_string_view< _CharT, _Traits > __x, type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept -> decltype(__detail::__char_traits_cmp_cat< _Traits >(0))
- template<typename _CharT, typename _Traits>
  constexpr bool **std::operator==** (basic_string_view< _CharT, _Traits > __x, type_identity_t< basic_string_view< _CharT, _Traits > > __y) noexcept

Variables

- template<typename _CharT, typename _Traits>
  constexpr bool **std::ranges::enable_borrowed_range< basic_string_view< _CharT, _Traits > >**
- template<typename _CharT, typename _Traits>
  constexpr bool **std::ranges::enable_view< basic_string_view< _CharT, _Traits > >**

7.690.1 Detailed Description

This is a Standard C++ Library header.

7.691 syncstream File Reference

7.691.1 Detailed Description

This is a Standard C++ Library header.

7.692 experimental/system_error File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_SYSTEM_ERROR**

Variables

- template<typename _Tp>
  constexpr bool **std::experimental::is_error_code_enum_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_error_condition_enum_v**

7.692.1 Detailed Description

This is a TS C++ Library header.

7.693 system_error File Reference

Classes

- class std::error_category
- class std::error_code
- class std::error_condition
- struct std::hash< error_code >
- struct std::hash< error_condition >
- struct std::is_error_code_enum< _Tp >
- struct std::is_error_condition_enum< _Tp >
- class std::system_error

Namespaces

- namespace std

Functions

- const error_category & std::generic_category () noexcept
- void **std::__adl_only::make_error_code** ()=delete
- void **std::__adl_only::make_error_condition** ()=delete
- const error_category & std::system_category () noexcept

Variables

- template<typename _Tp>
  constexpr bool **std::is_error_code_enum_v**
- template<typename _Tp>
  constexpr bool **std::is_error_condition_enum_v**

7.693.1 Detailed Description

This is a Standard C++ Library header.

7.694 text_encoding File Reference

7.694.1 Detailed Description

This is a Standard C++ Library header.

7.695 tgmath.h File Reference

7.695.1 Detailed Description

This is a Standard C++ Library header.

7.696 thread File Reference

Namespaces

- namespace std

7.696.1 Detailed Description

This is a Standard C++ Library header.

7.697 bool_set File Reference

Classes

- class std::tr2::bool_set

Namespaces

- namespace std
- namespace std::tr2

Macros

- #define **_GLIBCXX_TR2_BOOL_SET**

Functions

- bool **std::tr2::certainly** (bool_set __b)
- bool **std::tr2::contains** (bool_set __s, bool_set __t)
- bool **std::tr2::equals** (bool_set __s, bool_set __t)
- bool **std::tr2::is_emptyset** (bool_set __b)
- bool **std::tr2::is_indeterminate** (bool_set __b)
- bool **std::tr2::is_singleton** (bool_set __b)
- bool_set **std::tr2::operator!=** (bool __s, bool_set __t)
- bool_set **std::tr2::operator!=** (bool_set __s, bool __t)
- bool_set **std::tr2::operator!=** (bool_set __s, bool_set __t)
- bool_set **std::tr2::operator&** (bool __s, bool_set __t)
- bool_set **std::tr2::operator&** (bool_set __s, bool __t)
- bool_set **std::tr2::operator==** (bool __s, bool_set __t)
- bool_set **std::tr2::operator==** (bool_set __s, bool __t)
- bool_set **std::tr2::operator^** (bool __s, bool_set __t)
- bool_set **std::tr2::operator^** (bool_set __s, bool __t)
- bool_set **std::tr2::operator|** (bool __s, bool_set __t)
- bool_set **std::tr2::operator|** (bool_set __s, bool __t)
- bool **std::tr2::possibly** (bool_set __b)
- bool_set **std::tr2::set_complement** (bool_set __b)
- bool_set **std::tr2::set_intersection** (bool __s, bool_set __t)
- bool_set **std::tr2::set_intersection** (bool_set __s, bool __t)
- bool_set **std::tr2::set_intersection** (bool_set __s, bool_set __t)
- bool_set **std::tr2::set_union** (bool __s, bool_set __t)
- bool_set **std::tr2::set_union** (bool_set __s, bool __t)
- bool_set **std::tr2::set_union** (bool_set __s, bool_set __t)

7.697.1 Detailed Description

This is a TR2 C++ Library header.

7.698 bool_set.tcc File Reference

Namespaces

- namespace std
- namespace std::tr2

7.698.1 Detailed Description

This is a TR2 C++ Library header.

7.699 dynamic_bitset File Reference

Classes

- struct std::tr2::__dynamic_bitset_base< _WordT, _Alloc >
- class std::tr2::dynamic_bitset< _WordT, _Alloc >
- class std::tr2::dynamic_bitset< _WordT, _Alloc >::reference

Namespaces

- namespace std
- namespace std::tr2

Macros

- #define **_GLIBCXX_TR2_DYNAMIC_BITSET**

Functions

- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
  std::basic_ostream< _CharT, _Traits > & std::tr2::operator<< (std::basic_ostream< _CharT, _Traits > &__os, const dynamic_bitset< _WordT, _Alloc > &__x)

- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator!= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator<= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator> (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)
- template<typename _WordT, typename _Alloc>
  bool std::tr2::operator>= (const dynamic_bitset< _WordT, _Alloc > &__lhs, const dynamic_bitset< _WordT, _Alloc > &__rhs)

- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator& (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)

- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator- (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator^ (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)
- template<typename _WordT, typename _Alloc>
  dynamic_bitset< _WordT, _Alloc > std::tr2::operator| (const dynamic_bitset< _WordT, _Alloc > &__x, const dynamic_bitset< _WordT, _Alloc > &__y)

7.699.1 Detailed Description

This is a TR2 C++ Library header.

7.700 dynamic_bitset.tcc File Reference

Namespaces

- namespace std
- namespace std::tr2

Functions

- template<typename _CharT, typename _Traits, typename _WordT, typename _Alloc>
  std::basic_istream< _CharT, _Traits > & std::tr2::operator>> (std::basic_istream< _CharT, _Traits > &__is, dynamic_bitset< _WordT, _Alloc > &__x)

7.700.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <tr2/dynamic_bitset>.

7.701 experimental/tuple File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_tuple**
- #define **_GLIBCXX_EXPERIMENTAL_TUPLE**

Functions

- template<typename _Fn, typename _Tuple, std::size_t... _Idx>
  constexpr decltype(auto) **std::experimental::__apply_impl** (_Fn &&__f, _Tuple &&__t, std::index_sequence< _Idx... >)
- template<typename _Fn, typename _Tuple>
  constexpr decltype(auto) **std::experimental::apply** (_Fn &&__f, _Tuple &&__t)

Variables

- template<typename _Tp>
  constexpr size_t **std::experimental::tuple_size_v**

7.701.1 Detailed Description

This is a TS C++ Library header.

7.702 tuple File Reference

Classes

- class std::tuple< _Elements >
- class std::tuple< _T1, _T2 >
- struct std::tuple_element< __i, tuple< _Types... > >
- struct std::tuple_size< tuple< _Elements... > >
- struct std::uses_allocator< tuple< _Types... >, _Alloc >

Namespaces

- namespace std

Functions

- template<size_t __i, typename _Head, typename... _Tail>
  constexpr _Head & **std::__get_helper** (_Tuple_impl< __i, _Head, _Tail... > &__t) noexcept
- template<size_t __i, typename _Head, typename... _Tail>
  constexpr const _Head & **std::__get_helper** (const _Tuple_impl< __i, _Head, _Tail... > &__t) noexcept
- template<size_t __i, typename... _Types>
  __enable_if_t<(__i >=sizeof...(_Types))> **std::__get_helper** (const tuple< _Types... > &)=delete
- template<typename _Cat, typename _Tp, typename _Up, typename _IndexSeq>
  constexpr _Cat **std::__tuple_cmp** (const _Tp &__t, const _Up &__u, _IndexSeq __indices)
- template<typename... _Elements>
  constexpr tuple< _Elements &&... > std::forward_as_tuple (_Elements &&... __args) noexcept
- template<size_t __i, typename... _Elements>
  constexpr const __tuple_element_t< __i, tuple< _Elements... > > && std::get (const tuple< _Elements... > &&__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr const __tuple_element_t< __i, tuple< _Elements... > > & std::get (const tuple< _Elements... > &__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr __tuple_element_t< __i, tuple< _Elements... > > && std::get (tuple< _Elements... > &&__t) noexcept
- template<size_t __i, typename... _Elements>
  constexpr __tuple_element_t< __i, tuple< _Elements... > > & std::get (tuple< _Elements... > &__t) noexcept
- template<typename... _Elements>
  constexpr tuple< typename __decay_and_strip< _Elements >::__type... > std::make_tuple (_Elements &&... __args)
- template<typename... _Tps, typename... _Ups>
  constexpr common_comparison_category_t< __detail::__synth3way_t< _Tps, _Ups >... > **std::operator<=>** (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)
- template<typename... _Tps, typename... _Ups>
  constexpr bool **std::operator==** (const tuple< _Tps... > &__t, const tuple< _Ups... > &__u)
- template<typename... _Elements>
  constexpr enable_if<!__and_< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &, tuple< _Elements... > &)=delete
- template<typename... _Elements>
  constexpr enable_if< __and_< __is_swappable< _Elements >... >::value >::type std::swap (tuple< _Elements... > &__x, tuple< _Elements... > &__y) noexcept(noexcept(__x.swap(__y)))

- template<typename... _Elements>
  constexpr tuple< _Elements &... > std::tie (_Elements &... __args) noexcept
- template<typename... _UTypes>
  std::tuple (_UTypes...) -> tuple< _UTypes... >
- template<typename _Alloc, typename... _UTypes>
  std::tuple (allocator_arg_t, _Alloc, _UTypes...) -> tuple< _UTypes... >
- template<typename _Alloc, typename _T1, typename _T2>
  std::tuple (allocator_arg_t, _Alloc, pair< _T1, _T2 >) -> tuple< _T1, _T2 >
- template<typename _Alloc, typename... _UTypes>
  std::tuple (allocator_arg_t, _Alloc, tuple< _UTypes... >) -> tuple< _UTypes... >
- template<typename _T1, typename _T2>
  std::tuple (pair< _T1, _T2 >) -> tuple< _T1, _T2 >
- template<typename... _Tpls, typename = typename enable_if<__and_<__is_tuple_like<_Tpls>...>::value>::type>
  constexpr auto std::tuple_cat (_Tpls &&... __tpls) -> typename __tuple_cat_result< _Tpls... >::__type

Variables

- template<typename... _Types>
  constexpr size_t **std::tuple_size_v**< **const tuple**< **_Types...** > >
- template<typename... _Types>
  constexpr size_t **std::tuple_size_v**< **tuple**< **_Types...** > >

7.702.1 Detailed Description

This is a Standard C++ Library header.

7.703 experimental/type_traits File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_TYPE_TRAITS**
- #define **__cpp_lib_experimental_type_trait_variable_templates**
- template<typename _Tp>
  constexpr size_t **std::experimental::alignment_of_v**
- template<typename _Tp, unsigned _Idx = 0>
  constexpr size_t **std::experimental::extent_v**
- template<typename _Tp>
  constexpr bool **std::experimental::has_virtual_destructor_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_abstract_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_arithmetic_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_array_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_assignable_v**
- template<typename _Base, typename _Derived>
  constexpr bool **std::experimental::is_base_of_v**

- template<typename _Tp>
  constexpr bool **std::experimental::is_class_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_compound_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_const_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **std::experimental::is_constructible_v**
- template<typename _From, typename _To>
  constexpr bool **std::experimental::is_convertible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_default_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_destructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_empty_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_enum_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_final_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_floating_point_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_function_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_fundamental_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_integral_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_literal_type_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_lvalue_reference_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_member_function_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_member_object_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_member_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_move_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_move_constructible_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_nothrow_assignable_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **std::experimental::is_nothrow_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_copy_assignable_v**

- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_default_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_destructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_move_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_nothrow_move_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_null_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_object_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_pod_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_pointer_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_polymorphic_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_reference_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_rvalue_reference_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_same_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_same_v< _Tp, _Tp >**
- template<typename _Tp>
  constexpr bool **std::experimental::is_scalar_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_signed_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_standard_layout_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivial_v**
- template<typename _Tp, typename _Up>
  constexpr bool **std::experimental::is_trivially_assignable_v**
- template<typename _Tp, typename... _Args>
  constexpr bool **std::experimental::is_trivially_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_copy_assignable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_copy_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_copyable_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_default_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_destructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_move_assignable_v**

- template<typename _Tp>
  constexpr bool **std::experimental::is_trivially_move_constructible_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_union_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_unsigned_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_void_v**
- template<typename _Tp>
  constexpr bool **std::experimental::is_volatile_v**
- template<typename _Tp>
  constexpr size_t **std::experimental::rank_v**

- #define __cpp_lib_experimental_detect
- template<typename _Default, template< typename... > class _Op, typename... _Args>
  using std::experimental::detected_or
- template<typename _Default, template< typename... > class _Op, typename... _Args>
  using std::experimental::detected_or_t
- template<template< typename... > class _Op, typename... _Args>
  using std::experimental::detected_t
- template<template< typename... > class _Op, typename... _Args>
  using std::experimental::is_detected
- template<typename _To, template< typename... > class _Op, typename... _Args>
  using std::experimental::is_detected_convertible
- template<typename _To, template< typename... > class _Op, typename... _Args>
  constexpr bool std::experimental::is_detected_convertible_v
- template<typename _Expected, template< typename... > class _Op, typename... _Args>
  using std::experimental::is_detected_exact
- template<typename _Expected, template< typename... > class _Op, typename... _Args>
  constexpr bool std::experimental::is_detected_exact_v
- template<template< typename... > class _Op, typename... _Args>
  constexpr bool std::experimental::is_detected_v
- template<typename...>
  using std::experimental::void_t

- #define **__cpp_lib_experimental_logical_traits**
- template<typename... _Bn>
  constexpr bool **std::experimental::conjunction_v**
- template<typename... _Bn>
  constexpr bool **std::experimental::disjunction_v**
- template<typename _Pp>
  constexpr bool **std::experimental::negation_v**

7.703.1 Detailed Description

This is a TS C++ Library header.
This header defines variable templates for the C++14 type traits.
Equivalent variable templates are defined in namespace `std` since C++17.

See also

variable_templates

Since

C++14

7.704 tr2/type_traits File Reference

Classes

- struct std::tr2::__reflection_typelist< _First, _Rest... >
- struct std::tr2::__reflection_typelist<>
- struct std::tr2::bases< _Tp >
- struct std::tr2::direct_bases< _Tp >

Namespaces

- namespace std
- namespace std::tr2

Macros

- #define **_GLIBCXX_TR2_TYPE_TRAITS**

7.704.1 Detailed Description

This is a TR2 C++ Library header.

7.705 type_traits File Reference

Classes

- struct std::add_const< _Tp >
- struct std::add_cv< _Tp >
- struct std::add_lvalue_reference< _Tp >
- struct std::add_pointer< _Tp >
- struct std::add_rvalue_reference< _Tp >
- struct std::add_volatile< _Tp >
- struct std::aligned_storage< _Len, _Align >
- struct std::aligned_union< _Len, _Types >
- struct std::alignment_of< _Tp >
- struct std::conditional< _Cond, _Iftrue, _Iffalse >
- struct std::enable_if< bool, _Tp >
- struct std::extent< typename, _Uint >
- struct std::has_virtual_destructor< _Tp >
- struct std::integral_constant< _Tp, __v >
- struct std::is_abstract< _Tp >
- struct std::is_arithmetic< _Tp >
- struct std::is_array< _Tp >
- struct std::is_assignable< _Tp, _Up >
- struct std::is_base_of< _Base, _Derived >
- struct std::is_class< _Tp >
- struct std::is_compound< _Tp >
- struct std::is_const< _Tp >
- struct std::is_constructible< _Tp, _Args >
- struct std::is_copy_assignable< _Tp >
- struct std::is_copy_constructible< _Tp >
- struct std::is_default_constructible< _Tp >
- struct std::is_destructible< _Tp >
- struct std::is_empty< _Tp >

- struct std::is_enum< _Tp >
- struct std::is_floating_point< _Tp >
- struct std::is_function< _Tp >
- struct std::is_fundamental< _Tp >
- struct std::is_integral< _Tp >
- struct std::is_layout_compatible< _Tp, _Up >
- struct std::is_literal_type< _Tp >
- struct std::is_lvalue_reference< typename >
- struct std::is_member_function_pointer< _Tp >
- struct std::is_member_object_pointer< _Tp >
- struct std::is_member_pointer< _Tp >
- struct std::is_move_assignable< _Tp >
- struct std::is_move_constructible< _Tp >
- struct std::is_nothrow_assignable< _Tp, _Up >
- struct std::is_nothrow_constructible< _Tp, _Args >
- struct std::is_nothrow_copy_assignable< _Tp >
- struct std::is_nothrow_copy_constructible< _Tp >
- struct std::is_nothrow_default_constructible< _Tp >
- struct std::is_nothrow_destructible< _Tp >
- struct std::is_nothrow_move_assignable< _Tp >
- struct std::is_nothrow_move_constructible< _Tp >
- struct std::is_object< _Tp >
- struct std::is_pod< _Tp >
- struct std::is_pointer< _Tp >
- struct std::is_pointer_interconvertible_base_of< _Base, _Derived >
- struct std::is_polymorphic< _Tp >
- struct std::is_reference< _Tp >
- struct std::is_rvalue_reference< typename >
- struct std::is_same< _Tp, _Up >
- struct std::is_scalar< _Tp >
- struct std::is_signed< _Tp >
- struct std::is_standard_layout< _Tp >
- struct std::is_trivial< _Tp >
- struct std::is_trivially_assignable< _Tp, _Up >
- struct std::is_trivially_constructible< _Tp, _Args >
- struct std::is_trivially_copy_assignable< _Tp >
- struct std::is_trivially_copy_constructible< _Tp >
- struct std::is_trivially_copyable< _Tp >
- struct std::is_trivially_default_constructible< _Tp >
- struct std::is_trivially_destructible< _Tp >
- struct std::is_trivially_move_assignable< _Tp >
- struct std::is_trivially_move_constructible< _Tp >
- struct std::is_union< _Tp >
- struct std::is_unsigned< _Tp >
- struct std::is_void< _Tp >
- struct std::is_volatile< _Tp >
- struct std::make_signed< _Tp >
- struct std::make_unsigned< _Tp >
- struct std::rank< _Tp >
- struct std::remove_all_extents< _Tp >
- struct std::remove_const< _Tp >

- struct std::remove_cv< _Tp >
- struct std::remove_extent< _Tp >
- struct std::remove_pointer< _Tp >
- struct std::remove_reference< _Tp >
- struct std::remove_volatile< _Tp >
- struct std::underlying_type< _Tp >

Namespaces

- namespace std

Typedefs

- template<bool _Cond, typename _If, typename _Else>
  using **std::__conditional_t**
- template<bool _Cond, typename _Tp = void>
  using **std::__enable_if_t**
- template<typename _ToElementType, typename _FromElementType>
  using **std::__is_array_convertible**
- template<typename _Tp>
  using std::add_lvalue_reference_t
- template<typename _Tp>
  using std::add_pointer_t
- template<typename _Tp>
  using std::add_rvalue_reference_t
- template<size_t _Len, size_t _Align = __aligned_storage_default_alignment(_Len)>
  using std::aligned_storage_t
- template<size_t _Len, typename... _Types>
  using **std::aligned_union_t**
- template<typename... _Tp>
  using std::common_type_t
- template<bool _Cond, typename _Iftrue, typename _Iffalse>
  using std::conditional_t
- template<typename _Tp>
  using std::decay_t
- template<bool _Cond, typename _Tp = void>
  using std::enable_if_t
- using std::false_type
- template<typename _Tp>
  using std::make_signed_t
- template<typename _Tp>
  using std::make_unsigned_t
- template<typename _Tp>
  using std::remove_all_extents_t
- template<typename _Tp>
  using std::remove_extent_t
- template<typename _Tp>
  using std::remove_pointer_t
- template<typename _Tp>
  using std::remove_reference_t
- template<typename _Tp>
  using std::result_of_t
- using std::true_type

- template<typename _Tp>
  using std::underlying_type_t

Functions

- template<typename _Tp>
  auto std::declval () noexcept -> decltype(__declval< _Tp >(0))
- template<typename _Tp>
  constexpr _Require< __not_< __is_tuple_like< _Tp > >, is_move_constructible< _Tp >, is_move_assignable< _Tp > > std::swap (_Tp &, _Tp &) noexcept(__and_< is_nothrow_move_constructible< _Tp >, is_nothrow_move_assignable< _Tp > >::value)
- template<typename _Tp, size_t _Nm>
  constexpr __enable_if_t< __is_swappable< _Tp >::value > std::swap (_Tp(&__a)[_Nm], _Tp(&__b)[_Nm]↩
) noexcept(__is_nothrow_swappable< _Tp >::value)

Variables

- template<typename _Tp, typename... _Args>
  constexpr bool **std::__is_nothrow_new_constructible**
- template<size_t _Len, typename... _Types>
  const size_t **std::aligned_union< _Len, _Types... >::alignment_value**
- template<typename... _Tp>
  using std::common_reference_t
- template<typename _S1, typename _S2, typename _M1, typename _M2>
  constexpr bool std::is_corresponding_member (_M1 _S1::*__m1, _M2 _S2::*__m2) noexcept
- template<typename _Tp, typename _Up>
  constexpr bool std::is_layout_compatible_v
- template<typename _Base, typename _Derived>
  constexpr bool std::is_pointer_interconvertible_base_of_v
- template<typename _Tp, typename _Mem>
  constexpr bool std::is_pointer_interconvertible_with_class (_Mem _Tp::*__mp) noexcept

7.705.1 Detailed Description

This is a Standard C++ Library header.

7.706 typeindex File Reference

Classes

- struct std::hash< type_index >
- struct std::type_index

Namespaces

- namespace std

7.706.1 Detailed Description

This is a Standard C++ Library header.

7.707 debug/unordered_map File Reference

Classes

- class std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >
- class std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_UNORDERED_MAP**

Functions

- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  bool **std::__debug::operator==** (const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  bool **std::__debug::operator==** (const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, const unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y)
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  void **std::__debug::swap** (unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Key, typename _Tp, typename _Hash, typename _Pred, typename _Alloc>
  void **std::__debug::swap** (unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__x, unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_map (_InputIterator, _InputIterator, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_map (_InputIterator, _InputIterator, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_map (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >

- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Allocator) -> unordered_map< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type, _Hash, _Allocator) -> unordered_map< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_map (initializer_list< pair< _Key, _Tp > >, typename unordered_map< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_map< _Key, _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (_InputIterator, _InputIterator, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, hash< __iter_key_t< _InputIterator > >, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, equal_to< __iter_key_t< _InputIterator > >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<__iter_key_t<_InputIterator>>, typename _Pred = equal_to<__iter_key_t<_InputIterator>>, typename _Allocator = allocator<__iter_to_alloc_t<_InputIterator>>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (_InputIterator, _InputIterator, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< __iter_key_t< _InputIterator >, __iter_val_t< _InputIterator >, _Hash, _Pred, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (initializer_list< pair< _Key, _Tp > >, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Allocator) -> unordered_multimap< _Key, _Tp, hash< _Key >, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type, _Hash, _Allocator) -> unordered_multimap< _Key, _Tp, _Hash, equal_to< _Key >, _Allocator >
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>, typename _Allocator = allocator<pair<const _Key, _Tp>>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multimap (initializer_list< pair< _Key, _Tp > >, unordered_multimap< int, int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multimap< _Key, _Tp, _Hash, _Pred, _Allocator >

7.707.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.708 experimental/unordered_map File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_UNORDERED_MAP**

Typedefs

- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>
  using **std::experimental::fundamentals_v2::pmr::unordered_map**
- template<typename _Key, typename _Tp, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>
  using **std::experimental::fundamentals_v2::pmr::unordered_multimap**

Functions

- template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (unordered_map< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Tp, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (unordered_multimap< _Key, _Tp, _Hash, _CPred, _Alloc > &__cont, _↩ Predicate __pred)

7.708.1 Detailed Description

This is a TS C++ Library header.

7.709 unordered_map File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>
  using **std::pmr::unordered_map**
- template<typename _Key, typename _Tp, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>
  using **std::pmr::unordered_multimap**

7.709.1 Detailed Description

This is a Standard C++ Library header.

7.710 debug/unordered_set File Reference

Classes

- class std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >
- class std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >

Namespaces

- namespace std
- namespace std::__debug

Macros

- #define **_GLIBCXX_DEBUG_UNORDERED_SET**

Functions

- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  bool **std::__debug::operator==** (const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  bool **std::__debug::operator==** (const unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, const unordered_set< _Value, _Hash, _Pred, _Alloc > &__y)
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  void **std::__debug::swap** (unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__x, unordered_multiset< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _Value, typename _Hash, typename _Pred, typename _Alloc>
  void **std::__debug::swap** (unordered_set< _Value, _Hash, _Pred, _Alloc > &__x, unordered_set< _Value, _Hash, _Pred, _Alloc > &__y) noexcept(noexcept(__x.swap(__y)))
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (_InputIterator, _InputIterator, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (_InputIterator, _InputIterator, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >

- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (initializer_list< _Tp >, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Allocator) -> unordered_multiset< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type, _Hash, _Allocator) -> unordered_multiset< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_multiset (initializer_list< _Tp >, unordered_multiset< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_multiset< _Tp, _Hash, _Pred, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_set (_InputIterator, _InputIterator, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, hash< typename iterator_traits< _InputIterator >::value_type >, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash, typename _Allocator, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, equal_to< typename iterator_traits< _InputIterator >::value_type >, _Allocator >
- template<typename _InputIterator, typename _Hash = hash<typename iterator_traits<_InputIterator>::value_type>, typename _Pred = equal_to<typename iterator_traits<_InputIterator>::value_type>, typename _Allocator = allocator<typename iterator_traits<_InputIterator>::value_type>, typename = _RequireInputIter<_InputIterator>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_set (_InputIterator, _InputIterator, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< typename iterator_traits< _InputIterator >::value_type, _Hash, _Pred, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_set (initializer_list< _Tp >, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Allocator, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Allocator) -> unordered_set< _Tp, hash< _Tp >, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash, typename _Allocator, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type, _Hash, _Allocator) -> unordered_set< _Tp, _Hash, equal_to< _Tp >, _Allocator >
- template<typename _Tp, typename _Hash = hash<_Tp>, typename _Pred = equal_to<_Tp>, typename _Allocator = allocator<_Tp>, typename = _RequireNotAllocatorOrIntegral<_Hash>, typename = _RequireNotAllocator<_Pred>, typename = _RequireAllocator<_Allocator>>
  std::__debug::unordered_set (initializer_list< _Tp >, unordered_set< int >::size_type={}, _Hash=_Hash(), _Pred=_Pred(), _Allocator=_Allocator()) -> unordered_set< _Tp, _Hash, _Pred, _Allocator >

7.710.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.711 experimental/unordered_set File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_UNORDERED_SET**

Typedefs

- template<typename _Key, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>
  using **std::experimental::fundamentals_v2::pmr::unordered_multiset**
- template<typename _Key, typename _Hash = hash<_Key>, typename _Pred = equal_to<_Key>>
  using **std::experimental::fundamentals_v2::pmr::unordered_set**

Functions

- template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (unordered_multiset< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)
- template<typename _Key, typename _Hash, typename _CPred, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (unordered_set< _Key, _Hash, _CPred, _Alloc > &__cont, _Predicate __pred)

7.711.1 Detailed Description

This is a TS C++ Library header.

7.712 unordered_set File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>
  using **std::pmr::unordered_multiset**
- template<typename _Key, typename _Hash = std::hash<_Key>, typename _Pred = std::equal_to<_Key>>
  using **std::pmr::unordered_set**

7.712.1 Detailed Description

This is a Standard C++ Library header.

7.713 experimental/utility File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **_GLIBCXX_EXPERIMENTAL_UTILITY**

Typedefs

- using **std::experimental::erased_type**

7.713.1 Detailed Description

This is a TS C++ Library header.

7.714 utility File Reference

Namespaces

- namespace std

7.714.1 Detailed Description

This is a Standard C++ Library header.

7.715 valarray File Reference

Classes

- class std::valarray< _Tp >

Namespaces

- namespace std
- namespace std::__detail

Functions

- template<typename _Tp, size_t _Nm>
  std::valarray (const _Tp(&)[_Nm], size_t) -> valarray< _Tp >

7.715.1 Detailed Description

This is a Standard C++ Library header.

7.716 variant File Reference

7.716.1 Detailed Description

This is the <variant> C++ Library header.

7.717 debug/vector File Reference

Classes

- class __gnu_debug::_Safe_vector< _SafeSequence, _BaseSequence >
- struct std::hash< __debug::vector< bool, _Alloc > >
- class std::__debug::vector< _Tp, _Allocator >

Namespaces

- namespace __gnu_debug
- namespace std
- namespace std::__debug
- namespace std::__detail

Macros

- #define **_GLIBCXX_DEBUG_VECTOR**

Functions

- template<typename _Tp, typename _Alloc>
  constexpr __detail::__synth3way_t< _Tp > **std::__debug::operator<=>** (const vector< _Tp, _Alloc > &__x, const vector< _Tp, _Alloc > &__y)
- template<typename _Tp, typename _Alloc>
  constexpr bool **std::__debug::operator==** (const vector< _Tp, _Alloc > &__lhs, const vector< _Tp, _Alloc > &__rhs)
- template<typename _Tp, typename _Alloc>
  constexpr void **std::__debug::swap** (vector< _Tp, _Alloc > &__lhs, vector< _Tp, _Alloc > &__rhs) noexcept(/↩ *conditional */)
- template<typename _InputIterator, typename _ValT = typename iterator_traits<_InputIterator>::value_type, typename _Allocator = allocator<_ValT>, typename = _RequireInputIter<_InputIterator>, typename = _RequireAllocator<_Allocator>>
  std::__debug::vector (_InputIterator, _InputIterator, _Allocator=_Allocator()) -> vector< _ValT, _Allocator >
- template<typename _Tp, typename _Allocator = allocator<_Tp>, typename = _RequireAllocator<_Allocator>>
  std::__debug::vector (size_t, _Tp, _Allocator=_Allocator()) -> vector< _Tp, _Allocator >

7.717.1 Detailed Description

This file is a GNU debug extension to the Standard C++ Library.

7.718 experimental/vector File Reference

Namespaces

- namespace std
- namespace std::experimental

Macros

- #define **__cpp_lib_experimental_erase_if**
- #define **_GLIBCXX_EXPERIMENTAL_VECTOR**

Typedefs

- template<typename _Tp>
  using **std::experimental::fundamentals_v2::pmr::vector**

Functions

- template<typename _Tp, typename _Alloc, typename _Up>
  void **std::experimental::erase** (vector< _Tp, _Alloc > &__cont, const _Up &__value)
- template<typename _Tp, typename _Alloc, typename _Predicate>
  void **std::experimental::erase_if** (vector< _Tp, _Alloc > &__cont, _Predicate __pred)

7.718.1 Detailed Description

This is a TS C++ Library header.

7.719 vector File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Tp>
  using **std::pmr::vector**

7.719.1 Detailed Description

This is a Standard C++ Library header.

7.720 atomic_word.h File Reference

Macros

- #define **_GLIBCXX_READ_MEM_BARRIER**
- #define **_GLIBCXX_WRITE_MEM_BARRIER**

Typedefs

- typedef int **_Atomic_word**

7.720.1 Detailed Description

This file is a GNU extension to the Standard C++ Library.

7.721 basic_file.h File Reference

Namespaces

- namespace std

7.721.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

7.722 c++allocator.h File Reference

Namespaces

- namespace std

Typedefs

- template<typename _Tp>
  using std::__allocator_base

7.722.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <memory>.

7.723 c++config.h File Reference

Namespaces

- namespace __gnu_cxx
- namespace std

Macros

- #define **__GLIBCXX__**
- #define **__glibcxx_assert**(cond)
- #define **__N**(msgid)
- #define **_GLIBCXX11_DEPRECATED**
- #define **_GLIBCXX11_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX11_USE_C99_MATH**
- #define **_GLIBCXX11_USE_C99_STDIO**
- #define **_GLIBCXX11_USE_C99_STDLIB**
- #define **_GLIBCXX11_USE_C99_WCHAR**
- #define **_GLIBCXX14_DEPRECATED**
- #define **_GLIBCXX14_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX17_DEPRECATED**
- #define **_GLIBCXX17_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX20_DEPRECATED**
- #define **_GLIBCXX20_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX23_DEPRECATED**
- #define **_GLIBCXX23_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX26_CONSTEXPR**
- #define **_GLIBCXX26_DEPRECATED**
- #define **_GLIBCXX26_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX98_USE_C99_COMPLEX**
- #define **_GLIBCXX98_USE_C99_MATH**
- #define **_GLIBCXX98_USE_C99_STDIO**
- #define **_GLIBCXX98_USE_C99_STDLIB**
- #define **_GLIBCXX98_USE_C99_WCHAR**
- #define **_GLIBCXX_ASSERT_FAIL**(_Condition)
- #define **_GLIBCXX_ASSERTIONS**
- #define **_GLIBCXX_ATOMIC_WORD_BUILTINS**
- #define **_GLIBCXX_AUTO_CAST**(X)
- #define **_GLIBCXX_BEGIN_EXTERN_C**
- #define **_GLIBCXX_BEGIN_INLINE_ABI_NAMESPACE**(X)
- #define **_GLIBCXX_BEGIN_NAMESPACE_ALGO**
- #define **_GLIBCXX_BEGIN_NAMESPACE_CONTAINER**
- #define **_GLIBCXX_BEGIN_NAMESPACE_CXX11**
- #define **_GLIBCXX_BEGIN_NAMESPACE_LDBL**
- #define **_GLIBCXX_BEGIN_NAMESPACE_LDBL_OR_CXX11**
- #define **_GLIBCXX_BEGIN_NAMESPACE_VERSION**
- #define **_GLIBCXX_CAN_ALIGNAS_DESTRUCTIVE_SIZE**

- #define **_GLIBCXX_DARWIN_USE_64_BIT_INODE**
- #define **_GLIBCXX_DEFAULT_ABI_TAG**
- #define **_GLIBCXX_DEPRECATED**
- #define **_GLIBCXX_DEPRECATED_SUGGEST**(ALT)
- #define **_GLIBCXX_DOXYGEN_ONLY**(X)
- #define **_GLIBCXX_END_EXTERN_C**
- #define **_GLIBCXX_END_INLINE_ABI_NAMESPACE**(X)
- #define **_GLIBCXX_END_NAMESPACE_ALGO**
- #define **_GLIBCXX_END_NAMESPACE_CONTAINER**
- #define **_GLIBCXX_END_NAMESPACE_CXX11**
- #define **_GLIBCXX_END_NAMESPACE_LDBL**
- #define **_GLIBCXX_END_NAMESPACE_LDBL_OR_CXX11**
- #define **_GLIBCXX_END_NAMESPACE_VERSION**
- #define **_GLIBCXX_EXTERN_TEMPLATE**
- #define **_GLIBCXX_EXTERN_TEMPLATE**
- #define **_GLIBCXX_FAST_MATH**
- #define **_GLIBCXX_FULLY_DYNAMIC_STRING**
- #define **_GLIBCXX_HAS_BUILTIN**(B)
- #define **_GLIBCXX_HAVE___CXA_THREAD_ATEXIT_IMPL**
- #define **_GLIBCXX_HAVE_ACOSF**
- #define **_GLIBCXX_HAVE_ACOSL**
- #define **_GLIBCXX_HAVE_ALIGNED_ALLOC**
- #define **_GLIBCXX_HAVE_ARC4RANDOM**
- #define **_GLIBCXX_HAVE_AS_SYMVER_DIRECTIVE**
- #define **_GLIBCXX_HAVE_ASINF**
- #define **_GLIBCXX_HAVE_ASINL**
- #define **_GLIBCXX_HAVE_AT_QUICK_EXIT**
- #define **_GLIBCXX_HAVE_ATAN2F**
- #define **_GLIBCXX_HAVE_ATAN2L**
- #define **_GLIBCXX_HAVE_ATANF**
- #define **_GLIBCXX_HAVE_ATANL**
- #define **_GLIBCXX_HAVE_ATOMIC_LOCK_POLICY**
- #define **_GLIBCXX_HAVE_ATTRIBUTE_VISIBILITY**
- #define **_GLIBCXX_HAVE_BUILTIN_HAS_UNIQ_OBJ_REP**
- #define **_GLIBCXX_HAVE_BUILTIN_IS_AGGREGATE**
- #define **_GLIBCXX_HAVE_BUILTIN_LAUNDER**
- #define **_GLIBCXX_HAVE_C99_FLT_EVAL_TYPES**
- #define **_GLIBCXX_HAVE_CEILF**
- #define **_GLIBCXX_HAVE_CEILL**
- #define **_GLIBCXX_HAVE_COSF**
- #define **_GLIBCXX_HAVE_COSHF**
- #define **_GLIBCXX_HAVE_COSHL**
- #define **_GLIBCXX_HAVE_COSL**
- #define **_GLIBCXX_HAVE_DECL_STRNLEN**
- #define **_GLIBCXX_HAVE_DIRFD**
- #define **_GLIBCXX_HAVE_EXCEPTION_PTR_SINCE_GCC46**
- #define **_GLIBCXX_HAVE_EXPF**
- #define **_GLIBCXX_HAVE_EXPL**
- #define **_GLIBCXX_HAVE_FABSF**
- #define **_GLIBCXX_HAVE_FABSL**
- #define **_GLIBCXX_HAVE_FDOPENDIR**

- #define **_GLIBCXX_HAVE_FINITE**
- #define **_GLIBCXX_HAVE_FINITEF**
- #define **_GLIBCXX_HAVE_FINITEL**
- #define **_GLIBCXX_HAVE_FLOORF**
- #define **_GLIBCXX_HAVE_FLOORL**
- #define **_GLIBCXX_HAVE_FMODF**
- #define **_GLIBCXX_HAVE_FMODL**
- #define **_GLIBCXX_HAVE_FREXPF**
- #define **_GLIBCXX_HAVE_FREXPL**
- #define **_GLIBCXX_HAVE_FWRITE_UNLOCKED**
- #define **_GLIBCXX_HAVE_GETENTROPY**
- #define **_GLIBCXX_HAVE_GETIPINFO**
- #define **_GLIBCXX_HAVE_GETS**
- #define **_GLIBCXX_HAVE_HYPOT**
- #define **_GLIBCXX_HAVE_HYPOTF**
- #define **_GLIBCXX_HAVE_HYPOTL**
- #define **_GLIBCXX_HAVE_ICONV**
- #define **_GLIBCXX_HAVE_IS_CONSTANT_EVALUATED**
- #define **_GLIBCXX_HAVE_ISINF**
- #define **_GLIBCXX_HAVE_ISINFF**
- #define **_GLIBCXX_HAVE_ISINFL**
- #define **_GLIBCXX_HAVE_ISNAN**
- #define **_GLIBCXX_HAVE_ISNANF**
- #define **_GLIBCXX_HAVE_ISNANL**
- #define **_GLIBCXX_HAVE_ISWBLANK**
- #define **_GLIBCXX_HAVE_LC_MESSAGES**
- #define **_GLIBCXX_HAVE_LDEXPF**
- #define **_GLIBCXX_HAVE_LDEXPL**
- #define **_GLIBCXX_HAVE_LIMIT_AS**
- #define **_GLIBCXX_HAVE_LIMIT_DATA**
- #define **_GLIBCXX_HAVE_LIMIT_FSIZE**
- #define **_GLIBCXX_HAVE_LIMIT_RSS**
- #define **_GLIBCXX_HAVE_LIMIT_VMEM**
- #define **_GLIBCXX_HAVE_LINK**
- #define **_GLIBCXX_HAVE_LINUX_FUTEX**
- #define **_GLIBCXX_HAVE_LOG10F**
- #define **_GLIBCXX_HAVE_LOG10L**
- #define **_GLIBCXX_HAVE_LOGF**
- #define **_GLIBCXX_HAVE_LOGL**
- #define **_GLIBCXX_HAVE_LSEEK**
- #define **_GLIBCXX_HAVE_MBSTATE_T**
- #define **_GLIBCXX_HAVE_MEMALIGN**
- #define **_GLIBCXX_HAVE_MODFF**
- #define **_GLIBCXX_HAVE_MODFL**
- #define **_GLIBCXX_HAVE_O_NONBLOCK**
- #define **_GLIBCXX_HAVE_OPENAT**
- #define **_GLIBCXX_HAVE_POLL**
- #define **_GLIBCXX_HAVE_POSIX_MEMALIGN**
- #define **_GLIBCXX_HAVE_POWF**
- #define **_GLIBCXX_HAVE_POWL**
- #define **_GLIBCXX_HAVE_QUICK_EXIT**

- #define **_GLIBCXX_HAVE_READLINK**
- #define **_GLIBCXX_HAVE_S_ISREG**
- #define **_GLIBCXX_HAVE_SECURE_GETENV**
- #define **_GLIBCXX_HAVE_SETENV**
- #define **_GLIBCXX_HAVE_SINCOS**
- #define **_GLIBCXX_HAVE_SINCOSF**
- #define **_GLIBCXX_HAVE_SINCOSL**
- #define **_GLIBCXX_HAVE_SINF**
- #define **_GLIBCXX_HAVE_SINHF**
- #define **_GLIBCXX_HAVE_SINHL**
- #define **_GLIBCXX_HAVE_SINL**
- #define **_GLIBCXX_HAVE_SOCKATMARK**
- #define **_GLIBCXX_HAVE_SQRTF**
- #define **_GLIBCXX_HAVE_SQRTL**
- #define **_GLIBCXX_HAVE_STACKTRACE**
- #define **_GLIBCXX_HAVE_STRERROR_L**
- #define **_GLIBCXX_HAVE_STRERROR_R**
- #define **_GLIBCXX_HAVE_STRTOF**
- #define **_GLIBCXX_HAVE_STRTOLD**
- #define **_GLIBCXX_HAVE_STRUCT_DIRENT_D_TYPE**
- #define **_GLIBCXX_HAVE_STRXFRM_L**
- #define **_GLIBCXX_HAVE_SYMLINK**
- #define **_GLIBCXX_HAVE_SYMVER_SYMBOL_RENAMING_RUNTIME_SUPPORT**
- #define **_GLIBCXX_HAVE_TANF**
- #define **_GLIBCXX_HAVE_TANHF**
- #define **_GLIBCXX_HAVE_TANHL**
- #define **_GLIBCXX_HAVE_TANL**
- #define **_GLIBCXX_HAVE_TIMESPEC_GET**
- #define **_GLIBCXX_HAVE_TLS**
- #define **_GLIBCXX_HAVE_TRUNCATE**
- #define **_GLIBCXX_HAVE_UNLINKAT**
- #define **_GLIBCXX_HAVE_USELOCALE**
- #define **_GLIBCXX_HAVE_VFWSCANF**
- #define **_GLIBCXX_HAVE_VSWSCANF**
- #define **_GLIBCXX_HAVE_VWSCANF**
- #define **_GLIBCXX_HAVE_WCSTOF**
- #define **_GLIBCXX_HAVE_WRITEV**
- #define **_GLIBCXX_HOSTED**
- #define **_GLIBCXX_ICONV_CONST**
- #define **_GLIBCXX_INLINE_VERSION**
- #define **_GLIBCXX_LT_OBJDIR**
- #define **_GLIBCXX_MANGLE_SIZE_T**
- #define **_GLIBCXX_NAMESPACE_CXX11**
- #define **_GLIBCXX_NAMESPACE_LDBL**
- #define **_GLIBCXX_NAMESPACE_LDBL_OR_CXX11**
- #define **_GLIBCXX_NO_SPECIALIZATIONS**
- #define **_GLIBCXX_NODISCARD**
- #define **_GLIBCXX_NOEXCEPT_PARM**
- #define **_GLIBCXX_NOEXCEPT_QUAL**
- #define **_GLIBCXX_PSEUDO_VISIBILITY**(V)
- #define **_GLIBCXX_RELEASE**

- #define **_GLIBCXX_RES_LIMITS**
- #define **_GLIBCXX_STATIC_TZDATA**
- #define **_GLIBCXX_STD_A**
- #define **_GLIBCXX_STD_C**
- #define **_GLIBCXX_STDIO_EOF**
- #define **_GLIBCXX_STDIO_SEEK_CUR**
- #define **_GLIBCXX_STDIO_SEEK_END**
- #define **_GLIBCXX_SYMVER**
- #define **_GLIBCXX_SYMVER_GNU**
- #define **_GLIBCXX_SYNCHRONIZATION_HAPPENS_AFTER**(A)
- #define **_GLIBCXX_SYNCHRONIZATION_HAPPENS_BEFORE**(A)
- #define **_GLIBCXX_THROW_OR_ABORT**(_EXC)
- #define **_GLIBCXX_TXN_SAFE**
- #define **_GLIBCXX_TXN_SAFE_DYN**
- #define **_GLIBCXX_USE_ALLOCATOR_NEW**
- #define **_GLIBCXX_USE_BUILTIN_TRAIT**(BT)
- #define **_GLIBCXX_USE_C11_UCHAR_CXX11**
- #define **_GLIBCXX_USE_C99**
- #define **_GLIBCXX_USE_C99_COMPLEX**
- #define **_GLIBCXX_USE_C99_COMPLEX_TR1**
- #define **_GLIBCXX_USE_C99_CTYPE**
- #define **_GLIBCXX_USE_C99_CTYPE_TR1**
- #define **_GLIBCXX_USE_C99_FENV**
- #define **_GLIBCXX_USE_C99_FENV_TR1**
- #define **_GLIBCXX_USE_C99_INTTYPES**
- #define **_GLIBCXX_USE_C99_INTTYPES_TR1**
- #define **_GLIBCXX_USE_C99_INTTYPES_WCHAR_T**
- #define **_GLIBCXX_USE_C99_INTTYPES_WCHAR_T_TR1**
- #define **_GLIBCXX_USE_C99_MATH**
- #define **_GLIBCXX_USE_C99_MATH_FUNCS**
- #define **_GLIBCXX_USE_C99_MATH_TR1**
- #define **_GLIBCXX_USE_C99_STDINT**
- #define **_GLIBCXX_USE_C99_STDINT_TR1**
- #define **_GLIBCXX_USE_C99_STDIO**
- #define **_GLIBCXX_USE_C99_STDLIB**
- #define **_GLIBCXX_USE_C99_WCHAR**
- #define **_GLIBCXX_USE_CHDIR**
- #define **_GLIBCXX_USE_CHMOD**
- #define **_GLIBCXX_USE_CLOCK_MONOTONIC**
- #define **_GLIBCXX_USE_CLOCK_REALTIME**
- #define **_GLIBCXX_USE_CXX11_ABI**
- #define **_GLIBCXX_USE_DECIMAL_FLOAT**
- #define **_GLIBCXX_USE_DEV_RANDOM**
- #define **_GLIBCXX_USE_DUAL_ABI**
- #define **_GLIBCXX_USE_FCHMOD**
- #define **_GLIBCXX_USE_FCHMODAT**
- #define **_GLIBCXX_USE_FSEEKO_FTELLO**
- #define **_GLIBCXX_USE_GET_NPROCS**
- #define **_GLIBCXX_USE_GETCWD**
- #define **_GLIBCXX_USE_GETTIMEOFDAY**
- #define **_GLIBCXX_USE_GLIBC_STDIO_EXT**

- #define **_GLIBCXX_USE_INIT_PRIORITY_ATTRIBUTE**
- #define **_GLIBCXX_USE_LFS**
- #define **_GLIBCXX_USE_LONG_LONG**
- #define **_GLIBCXX_USE_LSTAT**
- #define **_GLIBCXX_USE_MKDIR**
- #define **_GLIBCXX_USE_NANOSLEEP**
- #define **_GLIBCXX_USE_NL_LANGINFO_L**
- #define **_GLIBCXX_USE_NLS**
- #define **_GLIBCXX_USE_PROC_SELF_STATUS**
- #define **_GLIBCXX_USE_PTHREAD_COND_CLOCKWAIT**
- #define **_GLIBCXX_USE_PTHREAD_MUTEX_CLOCKLOCK**
- #define **_GLIBCXX_USE_PTHREAD_RWLOCK_CLOCKLOCK**
- #define **_GLIBCXX_USE_RANDOM_TR1**
- #define **_GLIBCXX_USE_REALPATH**
- #define **_GLIBCXX_USE_SC_NPROCESSORS_ONLN**
- #define **_GLIBCXX_USE_SCHED_YIELD**
- #define **_GLIBCXX_USE_SENDFILE**
- #define **_GLIBCXX_USE_ST_MTIM**
- #define **_GLIBCXX_USE_STD_SPEC_FUNCS**
- #define **_GLIBCXX_USE_STDIO_LOCKING**
- #define **_GLIBCXX_USE_STRUCT_TM_TM_ZONE**
- #define **_GLIBCXX_USE_TMPNAM**
- #define **_GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_CXX20**
- #define **_GLIBCXX_USE_UCHAR_C8RTOMB_MBRTOC8_FCHAR8_T**
- #define **_GLIBCXX_USE_UTIME**
- #define **_GLIBCXX_USE_UTIMENSAT**
- #define **_GLIBCXX_USE_WCHAR_T**
- #define **_GLIBCXX_USE_WEAK_REF**
- #define **_GLIBCXX_VERBOSE**
- #define **_GLIBCXX_VERBOSE_ASSERT**
- #define **_GLIBCXX_VISIBILITY**(V)
- #define **_GLIBCXX_WEAK_DEFINITION**
- #define **_GLIBCXX_X86_RDRAND**
- #define **_GLIBCXX_X86_RDSEED**
- #define **_GLIBCXX_ZONEINFO_DIR**
- #define **_GTHREAD_USE_MUTEX_TIMEDLOCK**

Typedefs

- typedef decltype(nullptr) **std::nullptr_t**
- typedef __PTRDIFF_TYPE__ **std::ptrdiff_t**
- typedef __SIZE_TYPE__ **std::size_t**

Functions

- void **std::__glibcxx_assert_fail** (const char *__file, int __line, const char *__function, const char *__condition) noexcept
- constexpr bool **std::__is_constant_evaluated** () noexcept
- void **std::__terminate** () noexcept

7.723.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <version>.

7.724 c++io.h File Reference

Namespaces

- namespace std

Typedefs

- typedef FILE **std::__c_file**
- typedef __gthread_mutex_t **std::__c_lock**

7.724.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <ios>.

7.725 c++locale.h File Reference

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_C_LOCALE_GNU**
- #define **_GLIBCXX_NUM_CATEGORIES**

Typedefs

- typedef __locale_t **std::__c_locale**

Functions

- int **std::__convert_from_v** (const __c_locale &__cloc, char *__out, const int __size, const char *__fmt,...)

7.725.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.726 c++locale_internal.h File Reference

Namespaces

- namespace std

Functions

- Catalogs & **std::get_catalogs** ()

7.726.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.727 parallel/compatibility.h File Reference

Namespaces

- namespace __gnu_parallel

Functions

- template<typename _Tp>
  _Tp **__gnu_parallel::__add_omp** (volatile _Tp *__ptr, _Tp __addend)
- template<typename _Tp>
  bool **__gnu_parallel::__cas_omp** (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _Tp>
  bool __gnu_parallel::__compare_and_swap (volatile _Tp *__ptr, _Tp __comparand, _Tp __replacement)
- template<typename _Tp>
  _Tp __gnu_parallel::__fetch_and_add (volatile _Tp *__ptr, _Tp __addend)
- void __gnu_parallel::__yield ()

7.727.1 Detailed Description

Compatibility layer, mostly concerned with atomic operations.
This file is a GNU parallel extension to the Standard C++ Library and contains implementation details for the library's internal use.

7.728 x86_64-pc-linux-gnu/bits/compatibility.h File Reference

7.728.1 Detailed Description

This is an internal header file, included by other library sources. You should not attempt to use it directly.

7.729 cpu_defines.h File Reference

7.729.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

7.730 ctype_base.h File Reference

Classes

- struct std::ctype_base

Namespaces

- namespace std

7.730.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.731 ctype_inline.h File Reference

Namespaces

- namespace std

7.731.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.732 cxxabi_tweaks.h File Reference

Macros

- #define **_GLIBCXX_CXA_VEC_CTOR_RETURN**(x)
- #define **_GLIBCXX_GUARD_BIT**
- #define **_GLIBCXX_GUARD_PENDING_BIT**
- #define **_GLIBCXX_GUARD_SET**(x)
- #define **_GLIBCXX_GUARD_TEST**(x)
- #define **_GLIBCXX_GUARD_WAITING_BIT**

Typedefs

- typedef void **__cxxabiv1::__cxa_cdtor_return_type**
- typedef void **__cxxabiv1::__cxa_vec_ctor_return_type**

Variables

- __extension__ typedef int **__cxxabiv1::__guard**

7.732.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <cxxabi.h>.

7.733 error_constants.h File Reference

Namespaces

- namespace std

Enumerations

- enum class **errc** {
  address_family_not_supported , **address_in_use** , **address_not_available** , **already_connected** ,
  argument_list_too_long , **argument_out_of_domain** , **bad_address** , **bad_file_descriptor** ,
  broken_pipe , **connection_aborted** , **connection_already_in_progress** , **connection_refused** ,
  connection_reset , **cross_device_link** , **destination_address_required** , **device_or_resource_busy** ,
  directory_not_empty , **executable_format_error** , **file_exists** , **file_too_large** ,
  filename_too_long , **function_not_supported** , **host_unreachable** , **illegal_byte_sequence** ,
  inappropriate_io_control_operation , **interrupted** , **invalid_argument** , **invalid_seek** ,
  io_error , **is_a_directory** , **message_size** , **network_down** ,
  network_reset , **network_unreachable** , **no_buffer_space** , **no_child_process** ,
  no_lock_available , **no_message** , **no_protocol_option** , **no_space_on_device** ,
  no_such_device_or_address , **no_such_device** , **no_such_file_or_directory** , **no_such_process** ,

not_a_directory , **not_a_socket** , **not_connected** , **not_enough_memory** , **operation_in_progress** , **operation_not_permitted** , **operation_not_supported** , **operation_would_block** , **permission_denied** , **protocol_not_supported** , **read_only_file_system** , **resource_deadlock_would_occur** , **resource_unavailable_try_again** , **result_out_of_range** , **timed_out** , **too_many_files_open_in_system** , **too_many_files_open** , **too_many_links** , **too_many_symbolic_link_levels** , **wrong_protocol_type** }

7.733.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <system_error>.

7.734 extc++.h File Reference

7.734.1 Detailed Description

This is an implementation file for a precompiled header.

7.735 messages_members.h File Reference

Namespaces

- namespace std

7.735.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.736 opt_random.h File Reference

Namespaces

- namespace std

7.736.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <random>.

7.737 os_defines.h File Reference

Macros

- #define **__NO_CTYPE**
- #define **_GLIBCXX_MAY_HAVE___CXA_THREAD_ATEXIT_IMPL**
- #define **_GLIBCXX_NO_OBSOLETE_ISINF_ISNAN_DYNAMIC**

7.737.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <iosfwd>.

7.738 stdc++.h File Reference

7.738.1 Detailed Description

This is an implementation file for a precompiled header.

7.739 stdtr1c++.h File Reference

7.739.1 Detailed Description

This is an implementation file for a precompiled header.

7.740 time_members.h File Reference

Namespaces

- namespace std

7.740.1 Detailed Description

This is an internal header file, included by other library headers. Do not attempt to use it directly. Instead, include <locale>.

7.741 compare File Reference

Classes

- struct std::compare_three_way_result< _Tp, _Up >
- struct std::type_order< _Tp, _Up >

Namespaces

- namespace std
- namespace std::__detail

Typedefs

- template<typename _Tp, typename _Up>
  using **std::__detail::__cmp3way_res_t**
- template<typename... _Ts>
  using **std::common_comparison_category_t**
- template<typename _Tp, typename _Up = _Tp>
  using std::compare_three_way_result_t
- using **std::__cmp_cat::type**

Enumerations

- enum class **_Ord** : type { **equivalent** , **less** , **greater** , **unordered** }

Functions

- template<typename... _Ts>
  constexpr auto **std::__detail::__common_cmp_cat** ()
- template<typename _Ordering>
  constexpr _Ordering **std::__cmp_cat::__make** (_Ord __o) noexcept
- template<typename _Ordering>
  constexpr _Ord **std::__cmp_cat::__ord** (_Ordering __o) noexcept

- constexpr bool **std::is_eq** (partial_ordering __cmp) noexcept
- constexpr bool **std::is_gt** (partial_ordering __cmp) noexcept
- constexpr bool **std::is_gteq** (partial_ordering __cmp) noexcept
- constexpr bool **std::is_lt** (partial_ordering __cmp) noexcept
- constexpr bool **std::is_lteq** (partial_ordering __cmp) noexcept
- constexpr bool **std::is_neq** (partial_ordering __cmp) noexcept

Variables

- template<typename _Tp>
  constexpr unsigned **std::__detail::__cmp_cat_id**
- template<> constexpr unsigned **std::__detail::__cmp_cat_id**< **partial_ordering** >
- template<> constexpr unsigned **std::__detail::__cmp_cat_id**< **strong_ordering** >
- template<> constexpr unsigned **std::__detail::__cmp_cat_id**< **weak_ordering** >
- constexpr __compare::_Partial_fallback **std::compare_partial_order_fallback**
- constexpr __compare::_Strong_fallback **std::compare_strong_order_fallback**
- constexpr __compare::_Weak_fallback **std::compare_weak_order_fallback**
- constexpr __compare::_Partial_order **std::partial_order**
- constexpr __compare::_Strong_order **std::strong_order**
- template<typename _Tp, typename _Up>
  constexpr strong_ordering std::type_order_v
- constexpr __compare::_Weak_order **std::weak_order**

7.741.1 Detailed Description

This is a Standard C++ Library header.

7.742 cxxabi.h File Reference

Classes

- class __gnu_cxx::recursive_init_error

Namespaces

- namespace __gnu_cxx
- namespace abi

Typedefs

- typedef __cxa_cdtor_return_type(* **__cxxabiv1::__cxa_cdtor_type**) (void *)

Functions

- __cxa_dependent_exception * **__cxxabiv1::__cxa_allocate_dependent_exception** () noexcept
- int **__cxxabiv1::__cxa_atexit** (void(*)(void *), void *, void *) noexcept
- void **__cxxabiv1::__cxa_bad_cast** ()
- void **__cxxabiv1::__cxa_bad_typeid** ()
- void * **__cxxabiv1::__cxa_begin_catch** (void *) noexcept
- std::type_info * **__cxxabiv1::__cxa_current_exception_type** () noexcept
- void **__cxxabiv1::__cxa_deleted_virtual** (void)
- char * __cxxabiv1::__cxa_demangle (const char *__mangled_name, char *__output_buffer, size_t *__length, int *__status)
- void **__cxxabiv1::__cxa_end_catch** ()

- void **__cxxabiv1::__cxa_finalize** (void *)
- void **__cxxabiv1::__cxa_free_dependent_exception** (__cxa_dependent_exception *) noexcept
- void **__cxxabiv1::__cxa_free_exception** (void *) noexcept
- void * **__cxxabiv1::__cxa_get_exception_ptr** (void *) noexcept
- __cxa_eh_globals * **__cxxabiv1::__cxa_get_globals** () noexcept
- __cxa_eh_globals * **__cxxabiv1::__cxa_get_globals_fast** () noexcept
- void **__cxxabiv1::__cxa_guard_abort** (__guard *) noexcept
- int **__cxxabiv1::__cxa_guard_acquire** (__guard *)
- void **__cxxabiv1::__cxa_guard_release** (__guard *) noexcept
- void **__cxxabiv1::__cxa_pure_virtual** (void)
- void **__cxxabiv1::__cxa_rethrow** ()
- int **__cxxabiv1::__cxa_thread_atexit** (void(*)(void *), void *, void *) noexcept
- void **__cxxabiv1::__cxa_throw** (void *, std::type_info *, void(*)(void *))
- void **__cxxabiv1::__cxa_throw_bad_array_new_length** ()
- __cxa_vec_ctor_return_type **__cxxabiv1::__cxa_vec_cctor** (void *__dest_array, void *__src_array, size_t __element_count, size_t __element_size, __cxa_cdtor_return_type(*__constructor)(void *, void *), __cxa_cdtor_type __destructor)
- void **__cxxabiv1::__cxa_vec_cleanup** (void *__array_address, size_t __element_count, size_t __s, __cxa_cdtor_type __destructor) noexcept
- __cxa_vec_ctor_return_type **__cxxabiv1::__cxa_vec_ctor** (void *__array_address, size_t __element_count, size_t __element_size, __cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor)
- void **__cxxabiv1::__cxa_vec_delete** (void *__array_address, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __destructor)
- void **__cxxabiv1::__cxa_vec_delete2** (void *__array_address, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __destructor, void(*__dealloc)(void *))
- void **__cxxabiv1::__cxa_vec_delete3** (void *__array_address, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __destructor, void(*__dealloc)(void *, size_t))
- void **__cxxabiv1::__cxa_vec_dtor** (void *__array_address, size_t __element_count, size_t __element_size, __cxa_cdtor_type __destructor)
- void * **__cxxabiv1::__cxa_vec_new** (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor)
- void * **__cxxabiv1::__cxa_vec_new2** (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor, void *(*__alloc)(size_t), void(*__dealloc)(void *))
- void * **__cxxabiv1::__cxa_vec_new3** (size_t __element_count, size_t __element_size, size_t __padding_size, __cxa_cdtor_type __constructor, __cxa_cdtor_type __destructor, void *(*__alloc)(size_t), void(*__dealloc)(void *, size_t))
- void * **__cxxabiv1::__dynamic_cast** (const void *__src_ptr, const __class_type_info *__src_type, const __class_type_info *__dst_type, ptrdiff_t __src2dst)

7.742.1 Detailed Description

The header provides an interface to the C++ ABI.

7.742.2 Function Documentation

__cxa_demangle()

```
char * __cxxabiv1::__cxa_demangle (
            const char * __mangled_name,
            char * __output_buffer,
            size_t * __length,
            int * __status)
```

Demangling routine. ABI-mandated entry point in the C++ runtime library for demangling.

Parameters

__mangled_name	A NUL-terminated character string containing the name to be demangled.
__output_buffer	A region of memory, allocated with malloc, of *__length* bytes, into which the demangled name is stored. If *__output_buffer* is not long enough, it is expanded using realloc. *__output_buffer* may instead be null; in that case, the demangled name is placed in a region of memory allocated with malloc.
__length	If *__length* is non-null, the length of the buffer containing the demangled name is placed in *__length*.
__status	If *__status* is non-null, *__status* is set to one of the following values: 0: The demangling operation succeeded. -1: A memory allocation failure occurred. -2: *mangled_name* is not a valid name under the C++ ABI mangling rules. -3: One of the arguments is invalid.

Returns

A pointer to the start of the NUL-terminated demangled name, or a null pointer if the demangling fails. The caller is responsible for deallocating this memory using `free`.

The demangling is performed using the C++ ABI mangling rules, with GNU extensions. For example, this function is used in __gnu_cxx::__verbose_terminate_handler.
See https://gcc.gnu.org/onlinedocs/libstdc++/manual/ext_demangling.html for other examples of use.

Note

The same demangling functionality is available via libiberty (<libiberty/demangle.h> and libiberty.a) in GCC 3.1 and later, but that requires explicit installation (-enable-install-libiberty) and uses a different API, although the ABI is unchanged.

7.743 exception File Reference

Classes

- class std::bad_exception

Namespaces

- namespace __gnu_cxx
- namespace std

Typedefs

- typedef void(* std::terminate_handler) ()

Functions

- void __gnu_cxx::__verbose_terminate_handler ()
- terminate_handler std::get_terminate () noexcept
- terminate_handler std::set_terminate (terminate_handler) noexcept
- void std::terminate () noexcept
- bool std::uncaught_exception () noexcept

7.743.1 Detailed Description

This is a Standard C++ Library header.

7.744 initializer_list File Reference

Classes

- class std::initializer_list< _E >

Namespaces

- namespace std

7.744.1 Detailed Description

This is a Standard C++ Library header.

7.745 new File Reference

Classes

- struct std::destroying_delete_t

Namespaces

- namespace std

Macros

- #define **_GLIBCXX_PLACEMENT_CONSTEXPR**

Typedefs

- typedef void(* std::new_handler) ()

Enumerations

- enum class **align_val_t** : size_t

Functions

- new_handler std::get_new_handler () noexcept
- void **operator delete** (void *)
- void **operator delete** (void *, const std::nothrow_t &) noexcept
- void **operator delete** (void *, std::align_val_t) noexcept
- void **operator delete** (void *, std::align_val_t, const std::nothrow_t &) noexcept
- void **operator delete** (void *, std::size_t) noexcept
- void **operator delete** (void *, std::size_t, std::align_val_t) noexcept
- void **operator delete** (void *, void *) noexcept
- void **operator delete[ ]** (void *) noexcept
- void **operator delete[ ]** (void *, const std::nothrow_t &) noexcept
- void **operator delete[ ]** (void *, std::align_val_t) noexcept
- void **operator delete[ ]** (void *, std::align_val_t, const std::nothrow_t &) noexcept
- void **operator delete[ ]** (void *, std::size_t) noexcept

- void **operator delete[ ]** (void *, std::size_t, std::align_val_t) noexcept
- void **operator delete[ ]** (void *, void *) noexcept
- void * operator new (std::size_t)
- void * **operator new** (std::size_t, const std::nothrow_t &) noexcept
- void * **operator new** (std::size_t, std::align_val_t)
- void * **operator new** (std::size_t, std::align_val_t, const std::nothrow_t &) noexcept
- void * **operator new** (std::size_t, void *__p) noexcept
- void * **operator new[ ]** (std::size_t)
- void * **operator new[ ]** (std::size_t, const std::nothrow_t &) noexcept
- void * **operator new[ ]** (std::size_t, std::align_val_t)
- void * **operator new[ ]** (std::size_t, std::align_val_t, const std::nothrow_t &) noexcept
- void * **operator new[ ]** (std::size_t, void *__p) noexcept
- new_handler std::set_new_handler (new_handler) throw ()

Variables

- constexpr destroying_delete_t std::destroying_delete
- const nothrow_t **std::nothrow**

7.745.1 Detailed Description

This is a Standard C++ Library header.
The header `new` defines several functions to manage dynamic memory and handling memory allocation errors; see https://gcc.gnu.org/onlinedocs/libstdc++/manual/dynamic_memory.html for more.

7.745.2 Function Documentation

operator new()

```
void * operator new (
            std::size_t )  [nodiscard]
```

These are replaceable signatures:

- normal single new and delete (no arguments, throw `bad_alloc` on error)
- normal array new and delete (same)
- `nothrow` single new and delete (take a `nothrow` argument, return `NULL` on error)
- `nothrow` array new and delete (same)

Placement new and delete signatures (take a memory address argument, does nothing) may not be replaced by a user's program.

7.746 typeinfo File Reference

Classes

- class std::bad_cast
- class std::bad_typeid
- class std::type_info

Namespaces

- namespace std

Macros

- #define **__GXX_MERGED_TYPEINFO_NAMES**
- #define **__GXX_TYPEINFO_EQUALITY_INLINE**

7.746.1 Detailed Description

This is a Standard C++ Library header.

Index

_AlgorithmStrategy
__gnu_parallel, 431
_BALLOC_ALIGN_BYTES
bitmap_allocator.h, 3534
_BinIndex
__gnu_parallel, 431
_Bit_scan_forward
__gnu_cxx, 403
_CASable
__gnu_parallel, 431
_CASable_bits
__gnu_parallel, 472
_CASable_mask
__gnu_parallel, 472
_Construct
std, 590
_DRandomShufflingGlobalData
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 893
_Destroy
std, 590, 591
_Destroy_n
std, 591
_Distance_precision
__gnu_debug, 421
_FindAlgorithm
__gnu_parallel, 431
_Find_first
SGI, 156
_Find_next
SGI, 156
_GLIBCXX_BAL_QUICKSORT
features.h, 3655
_GLIBCXX_CALL
compiletime_settings.h, 3654
_GLIBCXX_DEBUG_VERIFY_AT_F
macros.h, 3511
_GLIBCXX_DEQUE_BUF_SIZE
stl_deque.h, 3413
_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS
features.h, 3655
_GLIBCXX_FIND_EQUAL_SPLIT
features.h, 3656
_GLIBCXX_FIND_GROWING_BLOCKS
features.h, 3656
_GLIBCXX_MERGESORT
features.h, 3656
_GLIBCXX_PARALLEL_ASSERTIONS
compiletime_settings.h, 3654
_GLIBCXX_PARALLEL_CONDITION
settings.h, 3672
_GLIBCXX_PARALLEL_LENGTH
multiway_merge.h, 3664
_GLIBCXX_QUICKSORT
features.h, 3656
_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1
compiletime_settings.h, 3654
_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB
compiletime_settings.h, 3654
_GLIBCXX_SCALE_DOWN_FPU
compiletime_settings.h, 3654
_GLIBCXX_TREE_DYNAMIC_BALANCING
features.h, 3656
_GLIBCXX_TREE_FULL_COPY
features.h, 3656
_GLIBCXX_TREE_INITIAL_SPLITTING
features.h, 3657
_GLIBCXX_VERBOSE_LEVEL
compiletime_settings.h, 3654
_GLIBCXX_VOLATILE
partition.h, 3668
queue.h, 3669
_GuardedIterator
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 907
_LoserTreeBase
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 940
_M_allocate_and_copy
std::vector< _Tp, _Alloc >, 3232
std::vector< bool, _Alloc >, 3249
_M_allocate_single_object
__gnu_cxx::bitmap_allocator< _Tp >, 1800
_M_attach
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 985, 986
__gnu_debug::_Safe_iterator_base, 994
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1000
__gnu_debug::_Safe_local_iterator_base, 1008
_M_attach_single
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 986
__gnu_debug::_Safe_iterator_base, 994
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1000, 1001
__gnu_debug::_Safe_local_iterator_base, 1008
_M_attached_to
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 986
__gnu_debug::_Safe_iterator_base, 994

__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1001
__gnu_debug::_Safe_local_iterator_base, 1008
_M_before_dereferenceable
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 986
_M_begin
__gnu_parallel::_Piece< _DifferenceTp >, 963
_M_bin_proc
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 893
_M_bins_begin
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >, 895
_M_buf
__gnu_cxx::enc_filebuf< _CharT >, 2127
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2956
std::basic_filebuf< _CharT, _Traits >, 1131
_M_buf_locale
__gnu_cxx::enc_filebuf< _CharT >, 2127
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2956
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2974
std::basic_filebuf< _CharT, _Traits >, 1131
std::basic_streambuf< _CharT, _Traits >, 1503
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1711
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3266
_M_buf_size
__gnu_cxx::enc_filebuf< _CharT >, 2127
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2956
std::basic_filebuf< _CharT, _Traits >, 1131
_M_can_compare
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 986
__gnu_debug::_Safe_iterator_base, 994
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1001
__gnu_debug::_Safe_local_iterator_base, 1008
_M_clear
__gnu_cxx::free_list, 2191
_M_comp
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 941
_M_const_iterators
__gnu_debug::_Safe_forward_list< _SafeSequence >, 981
__gnu_debug::_Safe_node_sequence< _Sequence >, 1013
__gnu_debug::_Safe_sequence< _Sequence >, 1016
__gnu_debug::_Safe_sequence_base, 1019
__gnu_debug::_Safe_unordered_container< _Container >, 1022
__gnu_debug::_Safe_unordered_container_base, 1025
_M_const_local_iterators
__gnu_debug::_Safe_unordered_container< _Container >, 1022
__gnu_debug::_Safe_unordered_container_base, 1025
_M_create_node
std::list< _Tp, _Alloc >, 2364
_M_create_pback
__gnu_cxx::enc_filebuf< _CharT >, 2115
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2940
std::basic_filebuf< _CharT, _Traits >, 1115
_M_deallocate_single_object
__gnu_cxx::bitmap_allocator< _Tp >, 1800
_M_dereferenceable
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 987
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1001
_M_destroy_pback
__gnu_cxx::enc_filebuf< _CharT >, 2115
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2940
std::basic_filebuf< _CharT, _Traits >, 1115
_M_detach
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 987
__gnu_debug::_Safe_iterator_base, 994
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1001
__gnu_debug::_Safe_local_iterator_base, 1009
_M_detach_all
__gnu_debug::_Safe_forward_list< _SafeSequence >, 981
__gnu_debug::_Safe_node_sequence< _Sequence >, 1012
__gnu_debug::_Safe_sequence< _Sequence >, 1015
__gnu_debug::_Safe_sequence_base, 1018
__gnu_debug::_Safe_unordered_container< _Container >, 1021
__gnu_debug::_Safe_unordered_container_base, 1024
_M_detach_single
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 987
__gnu_debug::_Safe_iterator_base, 994
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1001
__gnu_debug::_Safe_local_iterator_base, 1009
_M_detach_singular
__gnu_debug::_Safe_forward_list< _SafeSequence >, 981
__gnu_debug::_Safe_node_sequence< _Sequence >, 1012

__gnu_debug::_Safe_sequence< _Sequence >, 1015
__gnu_debug::_Safe_sequence_base, 1018
__gnu_debug::_Safe_unordered_container< _Container >, 1021
__gnu_debug::_Safe_unordered_container_base, 1024
_M_dist
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 893
_M_elements_leftover
__gnu_parallel::_QSBThreadLocal< _RAIter >, 972
_M_end
__gnu_parallel::_Piece< _DifferenceTp >, 963
_M_ext_buf
__gnu_cxx::enc_filebuf< _CharT >, 2128
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2956
std::basic_filebuf< _CharT, _Traits >, 1131
_M_ext_buf_size
__gnu_cxx::enc_filebuf< _CharT >, 2128
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2956
std::basic_filebuf< _CharT, _Traits >, 1131
_M_ext_next
__gnu_cxx::enc_filebuf< _CharT >, 2128
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2957
std::basic_filebuf< _CharT, _Traits >, 1131
_M_fill_initialize
std::deque< _Tp, _Alloc >, 2063
_M_finish_iterator
__gnu_parallel::__accumulate_selector< _It >, 746
__gnu_parallel::__adjacent_difference_selector< _It >, 747
__gnu_parallel::__count_if_selector< _It, _Diff >, 765
__gnu_parallel::__count_selector< _It, _Diff >, 766
__gnu_parallel::__fill_selector< _It >, 783
__gnu_parallel::__for_each_selector< _It >, 788
__gnu_parallel::__generate_selector< _It >, 789
__gnu_parallel::__generic_for_each_selector< _It >, 792
__gnu_parallel::__identity_selector< _It >, 793
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 795
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 820
__gnu_parallel::__replace_selector< _It, _Tp >, 822
__gnu_parallel::__transform1_selector< _It >, 824
__gnu_parallel::__transform2_selector< _It >, 825
_M_first
__gnu_parallel::_Job< _DifferenceTp >, 922
_M_first_insert
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 941
_M_gcount
std::basic_fstream< _CharT, _Traits >, 1178
std::basic_ifstream< _CharT, _Traits >, 1222
std::basic_iostream< _CharT, _Traits >, 1293
std::basic_istream< _CharT, _Traits >, 1332
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1373
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1756
_M_get
__gnu_cxx::free_list, 2191
_M_get_mutex
__gnu_debug::_Safe_forward_list< _SafeSequence >, 981
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 987
__gnu_debug::_Safe_iterator_base, 995
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1001
__gnu_debug::_Safe_local_iterator_base, 1009
__gnu_debug::_Safe_node_sequence< _Sequence >, 1012
__gnu_debug::_Safe_sequence< _Sequence >, 1015
__gnu_debug::_Safe_sequence_base, 1018
__gnu_debug::_Safe_unordered_container< _Container >, 1021
__gnu_debug::_Safe_unordered_container_base, 1024
_M_get_result
std::__basic_future< _Res >, 757
std::future< _Res >, 2204
std::future< _Res & >, 2205
std::future< void >, 2207
std::shared_future< _Res >, 2901
std::shared_future< _Res & >, 2902
std::shared_future< void >, 2904
_M_getloc
std::basic_fstream< _CharT, _Traits >, 1143
std::basic_ifstream< _CharT, _Traits >, 1194
std::basic_ios< _CharT, _Traits >, 1235
std::basic_iostream< _CharT, _Traits >, 1259
std::basic_istream< _CharT, _Traits >, 1306
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1346
std::basic_ofstream< _CharT, _Traits >, 1386
std::basic_ostream< _CharT, _Traits >, 1419
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1452
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1721
std::ios_base, 2292
_M_global
__gnu_parallel::_QSBThreadLocal< _RAIter >, 972
_M_in_beg

__gnu_cxx::enc_filebuf< _CharT >, 2128
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2957
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2974
std::basic_filebuf< _CharT, _Traits >, 1131
std::basic_streambuf< _CharT, _Traits >, 1503
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1711
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3267
_M_in_cur
__gnu_cxx::enc_filebuf< _CharT >, 2128
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2957
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2974
std::basic_filebuf< _CharT, _Traits >, 1131
std::basic_streambuf< _CharT, _Traits >, 1503
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1711
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3267
_M_in_end
__gnu_cxx::enc_filebuf< _CharT >, 2128
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2957
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2974
std::basic_filebuf< _CharT, _Traits >, 1132
std::basic_streambuf< _CharT, _Traits >, 1503
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1711
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3267
_M_in_same_bucket
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1001
_M_incrementable
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 987
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1002
_M_initial
__gnu_parallel::_QSBThreadLocal< _RAIter >, 972
_M_initialize_map
std::_Deque_base< _Tp, _Alloc >, 890
_M_insert
__gnu_cxx::free_list, 2191
_M_invalidate
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 987
__gnu_debug::_Safe_iterator_base, 995
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1002
__gnu_debug::_Safe_local_iterator_base, 1009
_M_invalidate_all
__gnu_debug::_Safe_sequence< _Sequence >, 1015
__gnu_debug::_Safe_sequence_base, 1019
__gnu_debug::_Safe_unordered_container< _Container >, 1021
__gnu_debug::_Safe_unordered_container_base, 1025
_M_invalidate_if
__gnu_debug::_Safe_forward_list< _SafeSequence >, 981
__gnu_debug::_Safe_node_sequence< _Sequence >, 1012
__gnu_debug::_Safe_sequence< _Sequence >, 1015
_M_is_before_begin
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 987
_M_is_begin
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 988
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1002
_M_is_beginnest
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 988
_M_is_end
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 988
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1002
_M_iterators
__gnu_debug::_Safe_forward_list< _SafeSequence >, 981
__gnu_debug::_Safe_node_sequence< _Sequence >, 1013
__gnu_debug::_Safe_sequence< _Sequence >, 1016
__gnu_debug::_Safe_sequence_base, 1019
__gnu_debug::_Safe_unordered_container< _Container >, 1022
__gnu_debug::_Safe_unordered_container_base, 1025
_M_key
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser, 934
_M_last
__gnu_parallel::_Job< _DifferenceTp >, 922
_M_leftover_parts
__gnu_parallel::_QSBThreadLocal< _RAIter >, 972
_M_load
__gnu_parallel::_Job< _DifferenceTp >, 922
_M_local_iterators
__gnu_debug::_Safe_unordered_container< _Container >, 1022
__gnu_debug::_Safe_unordered_container_base, 1025
_M_log_k
__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >, 936
__gnu_parallel::_LoserTree< false, _Tp, _Compare

>, 938
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 941
_M_losers
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 941
_M_mode
__gnu_cxx::enc_filebuf< _CharT >, 2128
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2957
std::basic_filebuf< _CharT, _Traits >, 1132
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1711
_M_new_elements_at_back
std::deque< _Tp, _Alloc >, 2063
_M_new_elements_at_front
std::deque< _Tp, _Alloc >, 2063
_M_next
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 991
__gnu_debug::_Safe_iterator_base, 995
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1005
__gnu_debug::_Safe_local_iterator_base, 1010
_M_num_bins
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 893
_M_num_bits
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 894
_M_num_threads
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >, 895
__gnu_parallel::_PMWMSSortingData< _RAIter >, 966
__gnu_parallel::_QSBThreadLocal< _RAIter >, 972
_M_offsets
__gnu_parallel::_PMWMSSortingData< _RAIter >, 966
_M_out_beg
__gnu_cxx::enc_filebuf< _CharT >, 2128
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2957
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2974
std::basic_filebuf< _CharT, _Traits >, 1132
std::basic_streambuf< _CharT, _Traits >, 1503
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1711
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3267
_M_out_cur
__gnu_cxx::enc_filebuf< _CharT >, 2129
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2957
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2974
std::basic_filebuf< _CharT, _Traits >, 1132
std::basic_streambuf< _CharT, _Traits >, 1504
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1711
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3267
_M_out_end
__gnu_cxx::enc_filebuf< _CharT >, 2129
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2957
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2974
std::basic_filebuf< _CharT, _Traits >, 1132
std::basic_streambuf< _CharT, _Traits >, 1504
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1711
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3267
_M_pback
__gnu_cxx::enc_filebuf< _CharT >, 2129
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2957
std::basic_filebuf< _CharT, _Traits >, 1132
_M_pback_cur_save
__gnu_cxx::enc_filebuf< _CharT >, 2129
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2958
std::basic_filebuf< _CharT, _Traits >, 1132
_M_pback_end_save
__gnu_cxx::enc_filebuf< _CharT >, 2129
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2958
std::basic_filebuf< _CharT, _Traits >, 1133
_M_pback_init
__gnu_cxx::enc_filebuf< _CharT >, 2129
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2958
std::basic_filebuf< _CharT, _Traits >, 1133
_M_pieces
__gnu_parallel::_PMWMSSortingData< _RAIter >, 966
_M_pop_back_aux
std::deque< _Tp, _Alloc >, 2063
_M_pop_front_aux
std::deque< _Tp, _Alloc >, 2063
_M_prior
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 991
__gnu_debug::_Safe_iterator_base, 995
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1005
__gnu_debug::_Safe_local_iterator_base, 1010
_M_push_back_aux
std::deque< _Tp, _Alloc >, 2063
_M_push_front_aux
std::deque< _Tp, _Alloc >, 2064
_M_range_check
std::deque< _Tp, _Alloc >, 2064
std::vector< _Tp, _Alloc >, 3232
_M_range_initialize
std::deque< _Tp, _Alloc >, 2064
_M_reading
__gnu_cxx::enc_filebuf< _CharT >, 2129
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2958
std::basic_filebuf< _CharT, _Traits >, 1133
_M_reallocate_map

std::deque< _Tp, _Alloc >, 2065
_M_requested_size
__gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >, 1040
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 2997
std::_Temporary_buffer< _ForwardIterator, _Tp >, 1042
_M_reserve_elements_at_back
std::deque< _Tp, _Alloc >, 2065
_M_reserve_elements_at_front
std::deque< _Tp, _Alloc >, 2065
_M_reserve_map_at_back
std::deque< _Tp, _Alloc >, 2065
_M_reserve_map_at_front
std::deque< _Tp, _Alloc >, 2065
_M_reset
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 988
__gnu_debug::_Safe_iterator_base, 995
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1002
__gnu_debug::_Safe_local_iterator_base, 1009
_M_revalidate_singular
__gnu_debug::_Safe_forward_list< _SafeSequence >, 981
__gnu_debug::_Safe_node_sequence< _Sequence >, 1012
__gnu_debug::_Safe_sequence< _Sequence >, 1015
__gnu_debug::_Safe_sequence_base, 1019
__gnu_debug::_Safe_unordered_container< _Container >, 1021
__gnu_debug::_Safe_unordered_container_base, 1025
_M_samples
__gnu_parallel::_PMWMSSortingData< _RAIter >, 966
_M_sd
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >, 895
_M_seed
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >, 895
_M_sequence
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 991
__gnu_debug::_Safe_iterator_base, 996
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1005
__gnu_debug::_Safe_local_iterator_base, 1010
_M_sequential_algorithm
__gnu_parallel::__adjacent_find_selector, 748
__gnu_parallel::__find_first_of_selector< _FIterator >, 784
__gnu_parallel::__find_if_selector, 785
__gnu_parallel::__mismatch_selector, 798
_M_set_buffer
__gnu_cxx::enc_filebuf< _CharT >, 2115
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2940
std::basic_filebuf< _CharT, _Traits >, 1115
_M_set_node
std::_Deque_iterator< _Tp, _Ref, _Ptr >, 892
_M_singular
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 988
__gnu_debug::_Safe_iterator_base, 995
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1002
__gnu_debug::_Safe_local_iterator_base, 1009
_M_source
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 894
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser, 934
__gnu_parallel::_PMWMSSortingData< _RAIter >, 966
_M_starts
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 894
__gnu_parallel::_PMWMSSortingData< _RAIter >, 966
_M_sup
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser, 934
_M_swap
__gnu_debug::_Safe_forward_list< _SafeSequence >, 981
__gnu_debug::_Safe_node_sequence< _Sequence >, 1012
__gnu_debug::_Safe_sequence< _Sequence >, 1015
__gnu_debug::_Safe_sequence_base, 1019
__gnu_debug::_Safe_unordered_container< _Container >, 1021
__gnu_debug::_Safe_unordered_container_base, 1025
_M_temporaries
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 894
_M_temporary
__gnu_parallel::_PMWMSSortingData< _RAIter >, 966
_M_transfer_from_if
__gnu_debug::_Safe_forward_list< _SafeSequence >, 981
__gnu_debug::_Safe_node_sequence< _Sequence >, 1012

__gnu_debug::_Safe_sequence< _Sequence >, 1016
_M_unlink
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 988
__gnu_debug::_Safe_iterator_base, 995
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1002
__gnu_debug::_Safe_local_iterator_base, 1009
_M_use_pointer
__gnu_parallel::_LoserTreeTraits< _Tp >, 948
_M_value_initialized
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 989
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1003
_M_version
__gnu_debug::_Safe_forward_list< _SafeSequence >, 982
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 991
__gnu_debug::_Safe_iterator_base, 996
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1005
__gnu_debug::_Safe_local_iterator_base, 1010
__gnu_debug::_Safe_node_sequence< _Sequence >, 1013
__gnu_debug::_Safe_sequence< _Sequence >, 1016
__gnu_debug::_Safe_sequence_base, 1019
__gnu_debug::_Safe_unordered_container< _Container >, 1022
__gnu_debug::_Safe_unordered_container_base, 1026
_M_w
std::_Base_bitset< 0 >, 882
std::_Base_bitset< _Nw >, 881
std::tr2::__dynamic_bitset_base< _WordT, _Alloc >, 782
_MultiwayMergeAlgorithm
__gnu_parallel, 431
_Opcode
Base and Implementation Classes, 278
_Parallelism
__gnu_parallel, 431
_PartialSumAlgorithm
__gnu_parallel, 431
_Piece
__gnu_parallel::_QSBThreadLocal< _RAIter >, 971
_PseudoSequence
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >, 969
_QSBThreadLocal
__gnu_parallel::_QSBThreadLocal< _RAIter >, 971
_RandomNumber
__gnu_parallel::_RandomNumber, 973
_RestrictedBoundedConcurrentQueue
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >, 977
_S_constant
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 989
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1003
_Safe_iterator
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 984, 985
_Safe_iterator_base
__gnu_debug::_Safe_iterator_base, 993
_Safe_local_iterator
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 999
_Safe_local_iterator_base
__gnu_debug::_Safe_local_iterator_base, 1007
_SequenceIndex
__gnu_parallel, 431
_SortAlgorithm
__gnu_parallel, 432
_SplittingAlgorithm
__gnu_parallel, 432
_Temporary_buffer
__gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >, 1040
std::_Temporary_buffer< _ForwardIterator, _Tp >, 1041
_ThreadIndex
__gnu_parallel, 431
_TokenT
std::__detail::_Scanner< _CharT >, 1029
_Unchecked_flip
SGI, 156
_Unchecked_reset
SGI, 157
_Unchecked_set
SGI, 157
_Unchecked_test
SGI, 157
__addressof
Utilities, 305
__allocator_base
Allocators, 333
__base
__gnu_debug, 421
__begin1_iterator
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 795
__begin2_iterator

__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 795
__bins_end
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >, 895
__bit_allocate
__gnu_cxx::__detail, 414
__bit_free
__gnu_cxx::__detail, 414
__calc_borders
__gnu_parallel, 432
__check_singular
__gnu_debug, 421
__check_singular_aux
__gnu_debug, 421
__check_string
__gnu_debug, 421
__compare_and_swap
__gnu_parallel, 432
__cpp_lib_experimental_detect
Detection idiom, 290
__ctype_type
std::basic_ios< _CharT, _Traits >, 1231
__cxa_demangle
cxxabi.h, 3727
__cxxabiv1::__forced_unwind, 788
__decode2
__gnu_parallel, 432
__delete_min_insert
__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >, 935
__gnu_parallel::_LoserTree< false, _Tp, _Compare >, 937
__determine_samples
__gnu_parallel, 433
__encode2
__gnu_parallel, 433
__equally_split
__gnu_parallel, 434
__equally_split_point
__gnu_parallel, 434
__fetch_and_add
__gnu_parallel, 435
__find_template
__gnu_parallel, 435–437
__for_each_template_random_access
__gnu_parallel, 437
__for_each_template_random_access_ed
__gnu_parallel, 438
__for_each_template_random_access_omp_loop
__gnu_parallel, 439
__for_each_template_random_access_omp_loop_static
__gnu_parallel, 440
__for_each_template_random_access_workstealing
__gnu_parallel, 441
__foreign_iterator_aux2
__gnu_debug, 422
__from_chars_alnum
std::__detail, 660
__from_chars_pow2_base
std::__detail, 660
__gen_two_uniform_ints
std, 590
__genrand_bits
__gnu_parallel::_RandomNumber, 974
__get_distance
__gnu_debug, 422
__get_min_source
__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >, 936
__gnu_parallel::_LoserTree< false, _Tp, _Compare >, 938
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 940
__get_num_threads
__gnu_parallel::balanced_quicksort_tag, 1108
__gnu_parallel::balanced_tag, 1109
__gnu_parallel::default_parallel_tag, 2050
__gnu_parallel::exact_tag, 2145
__gnu_parallel::multiway_mergesort_exact_tag, 2548
__gnu_parallel::multiway_mergesort_sampling_tag, 2549
__gnu_parallel::multiway_mergesort_tag, 2550
__gnu_parallel::omp_loop_static_tag, 2685
__gnu_parallel::omp_loop_tag, 2686
__gnu_parallel::parallel_tag, 2726
__gnu_parallel::quicksort_tag, 2789
__gnu_parallel::sampling_tag, 2869
__gnu_parallel::unbalanced_tag, 3090
__glibcxx_check_erase
macros.h, 3509
__glibcxx_check_erase_after
macros.h, 3509
__glibcxx_check_erase_range
macros.h, 3509
__glibcxx_check_erase_range_after
macros.h, 3509
__glibcxx_check_heap_pred
macros.h, 3510
__glibcxx_check_insert
macros.h, 3510
__glibcxx_check_insert_after
macros.h, 3510
__glibcxx_check_insert_range
macros.h, 3510
__glibcxx_check_insert_range_after
macros.h, 3510

__glibcxx_check_partitioned_lower
macros.h, 3511
__glibcxx_check_partitioned_lower_pred
macros.h, 3511
__glibcxx_check_partitioned_upper_pred
macros.h, 3511
__glibcxx_check_sorted_pred
macros.h, 3511
__gnu_cxx, 386
_Bit_scan_forward, 403
__int_traits, 403
__static_pointer_cast, 403
iota, 403
operator!=, 404
operator<, 407
operator<=, 408, 409
operator>, 410, 411
operator>=, 411, 412
operator+, 405, 406
operator==, 409, 410
swap, 413
__gnu_cxx::_Caster< _ToType >, 885
__gnu_cxx::_Char_types< _CharT >, 886
__gnu_cxx::_ExtPtr_allocator< _Tp >, 899
__gnu_cxx::_Invalid_type, 912
__gnu_cxx::_Pointer_adapter< _Storage_policy >, 967
__gnu_cxx::_Relative_pointer_impl< _Tp >, 974
__gnu_cxx::_Relative_pointer_impl< const _Tp >, 976
__gnu_cxx::_Std_pointer_impl< _Tp >, 1038
__gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >, 1039
_M_requested_size, 1040
_Temporary_buffer, 1040
begin, 1040
end, 1040
size, 1040
__gnu_cxx::_Unqualified_type< _Tp >, 1042
__gnu_cxx::__alloc_traits< _Alloc, typename >, 748
allocate, 751, 752
allocate_at_least, 752
const_void_pointer, 750
construct, 753
deallocate, 753, 754
destroy, 754
is_always_equal, 750
max_size, 754
propagate_on_container_copy_assignment, 750
propagate_on_container_move_assignment, 750
propagate_on_container_swap, 751
select_on_container_copy_construction, 755
void_pointer, 751
__gnu_cxx::__common_pool_policy< _PoolTp, _Thread >, 763
__gnu_cxx::__detail, 413
__bit_allocate, 414
__bit_free, 414
__num_bitmaps, 414
__num_blocks, 414
__gnu_cxx::__detail::_Bitmap_counter< _Tp >, 884
__gnu_cxx::__detail::_Ffit_finder< _Tp >, 900
__gnu_cxx::__detail::__mini_vector< _Tp >, 797
__gnu_cxx::__mt_alloc< _Tp, _Poolp >, 799
__gnu_cxx::__mt_alloc_base< _Tp >, 800
__gnu_cxx::__per_type_pool_policy< _Tp, _PoolTp, _Thread >, 810
__gnu_cxx::__pool< _Thread >, 810
__gnu_cxx::__pool< false >, 810
__gnu_cxx::__pool< true >, 811
__gnu_cxx::__pool_alloc< _Tp >, 812
__gnu_cxx::__pool_alloc_base, 813
__gnu_cxx::__pool_base, 814
__gnu_cxx::__rc_string_base< _CharT, _Traits, _Alloc >, 815
__gnu_cxx::__scoped_lock, 822
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 827
__versa_string, 830–833
~__versa_string, 833
append, 833–836
assign, 836–839
at, 840
back, 840, 841
begin, 841
c_str, 841
capacity, 841
cbegin, 841
cend, 842
clear, 842
compare, 842–844
copy, 845
crbegin, 846
crend, 846
data, 846
empty, 846
end, 846
erase, 847
find, 848, 849
find_first_not_of, 850, 851
find_first_of, 852, 853
find_last_not_of, 853–855
find_last_of, 855–857
front, 857
get_allocator, 857
insert, 857–862
length, 863
max_size, 863
npos, 879
operator+=, 863, 864

operator=, 865, 866
operator[], 866
pop_back, 867
push_back, 867
rbegin, 867
rend, 868
replace, 868–874
reserve, 875
resize, 875, 876
rfind, 876–878
shrink_to_fit, 878
size, 878
substr, 879
swap, 879
__gnu_cxx::allocator< _Tp >, 1048
__gnu_cxx::annotate_base, 1074
__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >, 1775
argument_type, 1775
result_type, 1775
__gnu_cxx::bitmap_allocator< _Tp >, 1799
_M_allocate_single_object, 1800
_M_deallocate_single_object, 1800
__gnu_cxx::char_traits< _CharT >, 1833
__gnu_cxx::character< _Value, _Int, _St >, 1841
__gnu_cxx::condition_base, 1910
__gnu_cxx::constant_binary_fun< _Result, _Arg1, _Arg2 >, 1919
__gnu_cxx::constant_unary_fun< _Result, _Argument >, 1920
__gnu_cxx::constant_void_fun< _Result >, 1921
__gnu_cxx::debug_allocator< _Alloc >, 2040
__gnu_cxx::enc_filebuf< _CharT >, 2112
_M_buf, 2127
_M_buf_locale, 2127
_M_buf_size, 2127
_M_create_pback, 2115
_M_destroy_pback, 2115
_M_ext_buf, 2128
_M_ext_buf_size, 2128
_M_ext_next, 2128
_M_in_beg, 2128
_M_in_cur, 2128
_M_in_end, 2128
_M_mode, 2128
_M_out_beg, 2128
_M_out_cur, 2129
_M_out_end, 2129
_M_pback, 2129
_M_pback_cur_save, 2129
_M_pback_end_save, 2129
_M_pback_init, 2129
_M_reading, 2129
_M_set_buffer, 2115
close, 2115
eback, 2115
egptr, 2115
epptr, 2116
gbump, 2116
getloc, 2116
gptr, 2116
imbue, 2117
in_avail, 2117
is_open, 2117
open, 2117
overflow, 2118
pbackfail, 2119
pbase, 2119
pbump, 2119
pptr, 2120
pubimbue, 2120
pubseekoff, 2120
pubseekpos, 2121
pubsetbuf, 2121
pubsync, 2121
sbumpc, 2121
seekoff, 2121
seekpos, 2122
setbuf, 2122
setg, 2122
setp, 2123
sgetc, 2123
sgetn, 2123
showmanyc, 2124
snextc, 2124
sputbackc, 2124
sputc, 2125
sputn, 2125
sungetc, 2125
sync, 2125
uflow, 2126
underflow, 2126
xsgetn, 2126
xsputn, 2127
__gnu_cxx::encoding_char_traits< _CharT >, 2130
__gnu_cxx::encoding_state, 2131
__gnu_cxx::equal_to< _Tp >, 2136
first_argument_type, 2136
result_type, 2136
second_argument_type, 2136
__gnu_cxx::forced_error, 2166
what, 2166
__gnu_cxx::free_list, 2190
_M_clear, 2191
_M_get, 2191
_M_insert, 2191
__gnu_cxx::hash_map< _Key, _Tp, _HashFn, _EqualKey, _Alloc >, 2257

__gnu_cxx::hash_multimap< _Key, _Tp, _HashFn, _EqualKey, _Alloc >, 2259
__gnu_cxx::hash_multiset< _Value, _HashFcn, _EqualKey, _Alloc >, 2260
__gnu_cxx::hash_set< _Value, _HashFcn, _EqualKey, _Alloc >, 2262
__gnu_cxx::limit_condition, 2349
__gnu_cxx::limit_condition::always_adjustor, 1074
__gnu_cxx::limit_condition::limit_adjustor, 2349
__gnu_cxx::limit_condition::never_adjustor, 2557
__gnu_cxx::malloc_allocator< _Tp >, 2408
__gnu_cxx::new_allocator< _Tp >, 2558
__gnu_cxx::pair< _T1, _T2 >, 2711
  explicit, 2714
  first, 2716
  first_type, 2713
  operator=, 2715, 2716
  pair, 2713
  second, 2716
  second_type, 2713
  swap, 2716
__gnu_cxx::project1st< _Arg1, _Arg2 >, 2780
  first_argument_type, 2781
  result_type, 2781
  second_argument_type, 2781
__gnu_cxx::project2nd< _Arg1, _Arg2 >, 2781
  first_argument_type, 2781
  result_type, 2781
  second_argument_type, 2782
__gnu_cxx::random_condition, 2790
__gnu_cxx::random_condition::always_adjustor, 1074
__gnu_cxx::random_condition::group_adjustor, 2232
__gnu_cxx::random_condition::never_adjustor, 2557
__gnu_cxx::rb_tree< _Key, _Value, _KeyOfValue, _Compare, _Alloc >, 2807
__gnu_cxx::recursive_init_error, 2821
  what, 2821
__gnu_cxx::rope< _CharT, _Alloc >, 2849
__gnu_cxx::select1st< _Pair >, 2872
  argument_type, 2873
  result_type, 2873
__gnu_cxx::select2nd< _Pair >, 2873
  argument_type, 2873
  result_type, 2873
__gnu_cxx::slist< _Tp, _Alloc >, 2925
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2936
  _M_buf, 2956
  _M_buf_locale, 2956
  _M_buf_size, 2956
  _M_create_pback, 2940
  _M_destroy_pback, 2940
  _M_ext_buf, 2956
  _M_ext_buf_size, 2956
  _M_ext_next, 2957
  _M_in_beg, 2957
  _M_in_cur, 2957
  _M_in_end, 2957
  _M_mode, 2957
  _M_out_beg, 2957
  _M_out_cur, 2957
  _M_out_end, 2957
  _M_pback, 2957
  _M_pback_cur_save, 2958
  _M_pback_end_save, 2958
  _M_pback_init, 2958
  _M_reading, 2958
  _M_set_buffer, 2940
  ~stdio_filebuf, 2940
  close, 2940
  eback, 2941
  egptr, 2941
  epptr, 2941
  fd, 2942
  file, 2942
  gbump, 2942
  getloc, 2942
  gptr, 2943
  imbue, 2943
  in_avail, 2943
  is_open, 2944
  open, 2944, 2945
  overflow, 2946
  pbackfail, 2946
  pbase, 2947
  pbump, 2947
  pptr, 2947
  pubimbue, 2947
  pubseekoff, 2948
  pubseekpos, 2948
  pubsetbuf, 2948
  pubsync, 2949
  sbumpc, 2949
  seekoff, 2949
  seekpos, 2949
  setbuf, 2950
  setg, 2950
  setp, 2951
  sgetc, 2951
  sgetn, 2951
  showmanyc, 2952
  snextc, 2952
  sputbackc, 2952
  sputc, 2953
  sputn, 2953
  stdio_filebuf, 2939
  sungetc, 2953
  sync, 2954
  uflow, 2954

underflow, 2954
xsgetn, 2955
xsputn, 2955
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2959
_M_buf_locale, 2974
_M_in_beg, 2974
_M_in_cur, 2974
_M_in_end, 2974
_M_out_beg, 2974
_M_out_cur, 2974
_M_out_end, 2974
eback, 2961
egptr, 2961
epptr, 2961
file, 2962
gbump, 2962
getloc, 2962
gptr, 2962
imbue, 2963
in_avail, 2963
overflow, 2963
pbackfail, 2964
pbase, 2964
pbump, 2965
pptr, 2965
pubimbue, 2965
pubseekoff, 2966
pubseekpos, 2966
pubsetbuf, 2966
pubsync, 2966
sbumpc, 2967
seekoff, 2967
seekpos, 2967
setbuf, 2967
setg, 2968
setp, 2968
sgetc, 2969
sgetn, 2969
showmanyc, 2969
snextc, 2970
sputbackc, 2970
sputc, 2970
sputn, 2971
sungetc, 2971
sync, 2971
uflow, 2972
underflow, 2972
xsgetn, 2972
xsputn, 2973
__gnu_cxx::subtractive_rng, 2991
argument_type, 2992
operator(), 2992
result_type, 2992
subtractive_rng, 2992
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 2995
_M_requested_size, 2997
~temporary_buffer, 2996
begin, 2997
end, 2997
size, 2997
temporary_buffer, 2996
__gnu_cxx::throw_allocator_base< _Tp, _Cond >, 3001
__gnu_cxx::throw_allocator_limit< _Tp >, 3002
__gnu_cxx::throw_allocator_random< _Tp >, 3004
__gnu_cxx::throw_value_base< _Cond >, 3005
__gnu_cxx::throw_value_limit, 3006
__gnu_cxx::throw_value_random, 3007
__gnu_cxx::typelist, 414
apply_generator, 415
__gnu_cxx::unary_compose< _Operation1, _Operation2 >, 3086
argument_type, 3087
result_type, 3087
__gnu_debug, 415
_Distance_precision, 421
__base, 421
__check_singular, 421
__check_singular_aux, 421
__check_string, 421
__foreign_iterator_aux2, 422
__get_distance, 422
__valid_range, 422
u16string, 421
u32string, 421
__gnu_debug::_After_nth_from< _Iterator >, 880
__gnu_debug::_BeforeBeginHelper< _Sequence >, 883
__gnu_debug::_Equal_to< _Type >, 896
__gnu_debug::_Not_equal_to< _Type >, 961
__gnu_debug::_Safe_container< _SafeContainer, _Alloc, _SafeBase, _IsCxx11AllocatorAware >, 978
__gnu_debug::_Safe_forward_list< _SafeSequence >, 980
_M_const_iterators, 981
_M_detach_all, 981
_M_detach_singular, 981
_M_get_mutex, 981
_M_invalidate_if, 981
_M_iterators, 981
_M_revalidate_singular, 981
_M_swap, 981
_M_transfer_from_if, 981
_M_version, 982
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 982
_M_attach, 985, 986
_M_attach_single, 986
_M_attached_to, 986

_M_before_dereferenceable, 986
_M_can_compare, 986
_M_dereferenceable, 987
_M_detach, 987
_M_detach_single, 987
_M_get_mutex, 987
_M_incrementable, 987
_M_invalidate, 987
_M_is_before_begin, 987
_M_is_begin, 988
_M_is_beginnest, 988
_M_is_end, 988
_M_next, 991
_M_prior, 991
_M_reset, 988
_M_sequence, 991
_M_singular, 988
_M_unlink, 988
_M_value_initialized, 989
_M_version, 991
_S_constant, 989
_Safe_iterator, 984, 985
base, 989
operator _Iterator, 989
operator++, 989, 990
operator->, 990
operator=, 990
operator*, 989
__gnu_debug::_Safe_iterator_base, 991
_M_attach, 994
_M_attach_single, 994
_M_attached_to, 994
_M_can_compare, 994
_M_detach, 994
_M_detach_single, 994
_M_get_mutex, 995
_M_invalidate, 995
_M_next, 995
_M_prior, 995
_M_reset, 995
_M_sequence, 996
_M_singular, 995
_M_unlink, 995
_M_version, 996
_Safe_iterator_base, 993
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 996
_M_attach, 1000
_M_attach_single, 1000, 1001
_M_attached_to, 1001
_M_can_compare, 1001
_M_dereferenceable, 1001
_M_detach, 1001
_M_detach_single, 1001
_M_get_mutex, 1001
_M_in_same_bucket, 1001
_M_incrementable, 1002
_M_invalidate, 1002
_M_is_begin, 1002
_M_is_end, 1002
_M_next, 1005
_M_prior, 1005
_M_reset, 1002
_M_sequence, 1005
_M_singular, 1002
_M_unlink, 1002
_M_value_initialized, 1003
_M_version, 1005
_S_constant, 1003
_Safe_local_iterator, 999
base, 1003
bucket, 1003
operator _Iterator, 1003
operator++, 1004
operator->, 1004
operator=, 1004
operator*, 1003
__gnu_debug::_Safe_local_iterator_base, 1005
_M_attach, 1008
_M_attach_single, 1008
_M_attached_to, 1008
_M_can_compare, 1008
_M_detach, 1009
_M_detach_single, 1009
_M_get_mutex, 1009
_M_invalidate, 1009
_M_next, 1010
_M_prior, 1010
_M_reset, 1009
_M_sequence, 1010
_M_singular, 1009
_M_unlink, 1009
_M_version, 1010
_Safe_local_iterator_base, 1007
__gnu_debug::_Safe_node_sequence< _Sequence >, 1010
_M_const_iterators, 1013
_M_detach_all, 1012
_M_detach_singular, 1012
_M_get_mutex, 1012
_M_invalidate_if, 1012
_M_iterators, 1013
_M_revalidate_singular, 1012
_M_swap, 1012
_M_transfer_from_if, 1012
_M_version, 1013
__gnu_debug::_Safe_sequence< _Sequence >, 1013
_M_const_iterators, 1016

_M_detach_all, 1015
_M_detach_singular, 1015
_M_get_mutex, 1015
_M_invalidate_all, 1015
_M_invalidate_if, 1015
_M_iterators, 1016
_M_revalidate_singular, 1015
_M_swap, 1015
_M_transfer_from_if, 1016
_M_version, 1016
__gnu_debug::_Safe_sequence_base, 1016
_M_const_iterators, 1019
_M_detach_all, 1018
_M_detach_singular, 1018
_M_get_mutex, 1018
_M_invalidate_all, 1019
_M_iterators, 1019
_M_revalidate_singular, 1019
_M_swap, 1019
_M_version, 1019
~_Safe_sequence_base, 1018
__gnu_debug::_Safe_unordered_container< _Container >, 1019
_M_const_iterators, 1022
_M_const_local_iterators, 1022
_M_detach_all, 1021
_M_detach_singular, 1021
_M_get_mutex, 1021
_M_invalidate_all, 1021
_M_iterators, 1022
_M_local_iterators, 1022
_M_revalidate_singular, 1021
_M_swap, 1021
_M_version, 1022
__gnu_debug::_Safe_unordered_container_base, 1022
_M_const_iterators, 1025
_M_const_local_iterators, 1025
_M_detach_all, 1024
_M_detach_singular, 1024
_M_get_mutex, 1024
_M_invalidate_all, 1025
_M_iterators, 1025
_M_local_iterators, 1025
_M_revalidate_singular, 1025
_M_swap, 1025
_M_version, 1026
~_Safe_unordered_container_base, 1024
__gnu_debug::_Safe_vector< _SafeSequence, _BaseSequence >, 1026
__gnu_debug::_Sequence_traits< _Sequence >, 1029
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1504
__resize_and_overwrite, 1511
append, 1511–1513
assign, 1514, 1515
at, 1516–1518
back, 1519
capacity, 1519
compare, 1519–1525
copy, 1526
data, 1526
empty, 1526
erase, 1526–1528
find, 1528–1533
find_first_not_of, 1533–1537
find_first_of, 1538–1542
find_last_not_of, 1543–1547
find_last_of, 1548–1552
front, 1552
get_allocator, 1553
insert, 1553–1559
length, 1560
max_size, 1560
npos, 1581
operator+=, 1560
operator[], 1560, 1561
replace, 1561–1571
reserve, 1572, 1573
resize, 1574
rfind, 1574–1578
size, 1579
substr, 1579
subview, 1580
swap, 1580
__gnu_debug::type_info, 3081
~type_info, 3082
before, 3082
name, 3082
__gnu_internal, 423
__gnu_parallel, 423
_AlgorithmStrategy, 431
_BinIndex, 431
_CASable, 431
_CASable_bits, 472
_CASable_mask, 472
_FindAlgorithm, 431
_MultiwayMergeAlgorithm, 431
_Parallelism, 431
_PartialSumAlgorithm, 431
_SequenceIndex, 431
_SortAlgorithm, 432
_SplittingAlgorithm, 432
_ThreadIndex, 431
__calc_borders, 432
__compare_and_swap, 432
__decode2, 432
__determine_samples, 433
__encode2, 433

__equally_split, 434
__equally_split_point, 434
__fetch_and_add, 435
__find_template, 435–437
__for_each_template_random_access, 437
__for_each_template_random_access_ed, 438
__for_each_template_random_access_omp_loop, 439
__for_each_template_random_access_omp_loop_static, 440
__for_each_template_random_access_workstealing, 441
__is_sorted, 441
__median_of_three_iterators, 442
__merge_advance, 442
__merge_advance_movc, 443
__merge_advance_usual, 444
__parallel_merge_advance, 444, 445
__parallel_nth_element, 445
__parallel_partial_sort, 446
__parallel_partial_sum, 446
__parallel_partial_sum_basecase, 447
__parallel_partial_sum_linear, 447
__parallel_partition, 448
__parallel_random_shuffle, 448
__parallel_random_shuffle_drs, 448
__parallel_random_shuffle_drs_pu, 449
__parallel_sort, 449–454
__parallel_sort_qs, 454
__parallel_sort_qs_conquer, 455
__parallel_sort_qs_divide, 455
__parallel_sort_qsb, 456
__parallel_unique_copy, 456, 457
__qsb_conquer, 457
__qsb_divide, 458
__qsb_local_sort_with_helping, 458
__random_number_pow2, 458
__rd_log2, 459
__round_up_to_pow2, 459
__search_template, 459
__sequential_multiway_merge, 460
__sequential_random_shuffle, 460
__shrink, 461
__shrink_and_double, 461
__yield, 462
list_partition, 462
max, 462
min, 463
multiseq_partition, 463
multiseq_selection, 463
multiway_merge, 464
multiway_merge_3_variant, 465
multiway_merge_4_variant, 466
multiway_merge_exact_splitting, 467
multiway_merge_loser_tree, 467
multiway_merge_loser_tree_sentinel, 467
multiway_merge_loser_tree_unguarded, 468
multiway_merge_sampling_splitting, 469
multiway_merge_sentinels, 469
parallel_balanced, 431
parallel_multiway_merge, 471
parallel_omp_loop, 431
parallel_omp_loop_static, 431
parallel_sort_mwms, 471
parallel_sort_mwms_pu, 472
parallel_taskqueue, 431
parallel_unbalanced, 431
sequential, 431
__gnu_parallel::_DRSSorterPU< _RAIter, _RandomNumberGenerator >, 894
_M_bins_begin, 895
_M_num_threads, 895
_M_sd, 895
_M_seed, 895
__bins_end, 895
__gnu_parallel::_DRandomShufflingGlobalData< _RAIter >, 892
_DRandomShufflingGlobalData, 893
_M_bin_proc, 893
_M_dist, 893
_M_num_bins, 893
_M_num_bits, 894
_M_source, 894
_M_starts, 894
_M_temporaries, 894
__gnu_parallel::_DummyReduct, 895
__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >, 896
first_argument_type, 897
result_type, 897
second_argument_type, 897
__gnu_parallel::_EqualTo< _T1, _T2 >, 897
first_argument_type, 898
result_type, 898
second_argument_type, 898
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 907
_GuardedIterator, 907
operator _RAIter, 908
operator<, 908
operator<=, 908
operator++, 908
operator*, 908
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 914
explicit, 917–919
first, 920
first_type, 917

operator<=>, 919
operator==, 919
second, 920
second_type, 917
swap, 919, 920
__gnu_parallel::_IteratorTriple< _Iterator1, _Iterator2, _Iterator3, _IteratorCategory >, 921
__gnu_parallel::_Job< _DifferenceTp >, 921
_M_first, 922
_M_last, 922
_M_load, 922
__gnu_parallel::_Less< _T1, _T2 >, 924
first_argument_type, 924
result_type, 924
second_argument_type, 924
__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >, 925
first_argument_type, 925
result_type, 926
second_argument_type, 926
__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >, 926
first_argument_type, 927
result_type, 927
second_argument_type, 927
__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >, 935
_M_log_k, 936
__delete_min_insert, 935
__get_min_source, 936
__insert_start, 936
__gnu_parallel::_LoserTree< false, _Tp, _Compare >, 936
_M_log_k, 938
__delete_min_insert, 937
__get_min_source, 938
__init_winner, 938
__insert_start, 938
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 939
_LoserTreeBase, 940
_M_comp, 941
_M_first_insert, 941
_M_log_k, 941
_M_losers, 941
__get_min_source, 940
__insert_start, 940
~_LoserTreeBase, 940
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >::_Loser, 934
_M_key, 934
_M_source, 934
_M_sup, 934
__gnu_parallel::_LoserTreePointer< __stable, _Tp, _Compare >, 941
__gnu_parallel::_LoserTreePointer< false, _Tp, _Compare >, 942
__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >, 943
__gnu_parallel::_LoserTreePointerBase< _Tp, _Compare >::_Loser, 934
__gnu_parallel::_LoserTreePointerUnguarded< __stable, _Tp, _Compare >, 944
__gnu_parallel::_LoserTreePointerUnguarded< false, _Tp, _Compare >, 945
__gnu_parallel::_LoserTreePointerUnguardedBase< _Tp, _Compare >, 946
__gnu_parallel::_LoserTreeTraits< _Tp >, 947
_M_use_pointer, 948
__gnu_parallel::_LoserTreeUnguarded< __stable, _Tp, _Compare >, 948
__gnu_parallel::_LoserTreeUnguarded< false, _Tp, _Compare >, 949
__gnu_parallel::_LoserTreeUnguardedBase< _Tp, _Compare >, 950
__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >, 952
first_argument_type, 953
result_type, 953
second_argument_type, 953
__gnu_parallel::_Nothing, 962
operator(), 963
__gnu_parallel::_PMWMSSortingData< _RAIter >, 965
_M_num_threads, 966
_M_offsets, 966
_M_pieces, 966
_M_samples, 966
_M_source, 966
_M_starts, 966
_M_temporary, 966
__gnu_parallel::_Piece< _DifferenceTp >, 963
_M_begin, 963
_M_end, 963
__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >, 964
first_argument_type, 965
result_type, 965
second_argument_type, 965
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >, 969
_PseudoSequence, 969
begin, 970
end, 970
__gnu_parallel::_PseudoSequenceIterator< _Tp, _DifferenceTp >, 970
__gnu_parallel::_QSBThreadLocal< _RAIter >, 971
_M_elements_leftover, 972
_M_global, 972
_M_initial, 972
_M_leftover_parts, 972
_M_num_threads, 972

_Piece, 971
_QSBThreadLocal, 971
__gnu_parallel::_RandomNumber, 973
_RandomNumber, 973
__genrand_bits, 974
operator(), 974
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >, 976
_RestrictedBoundedConcurrentQueue, 977
~_RestrictedBoundedConcurrentQueue, 977
pop_back, 977
pop_front, 977
push_front, 978
__gnu_parallel::_SamplingSorter< __stable, _RAIter, _StrictWeakOrdering >, 1027
__gnu_parallel::_SamplingSorter< false, _RAIter, _StrictWeakOrdering >, 1027
__gnu_parallel::_Settings, 1029
accumulate_minimal_n, 1031
adjacent_difference_minimal_n, 1031
cache_line_size, 1031
count_minimal_n, 1031
fill_minimal_n, 1032
find_increasing_factor, 1032
find_initial_block_size, 1032
find_maximum_block_size, 1032
find_scale_factor, 1032
find_sequential_search_size, 1032
for_each_minimal_n, 1032
generate_minimal_n, 1032
get, 1031
L1_cache_size, 1032
L2_cache_size, 1033
max_element_minimal_n, 1033
merge_minimal_n, 1033
merge_oversampling, 1033
min_element_minimal_n, 1033
multiway_merge_minimal_k, 1033
multiway_merge_minimal_n, 1033
multiway_merge_oversampling, 1033
nth_element_minimal_n, 1033
partial_sort_minimal_n, 1034
partial_sum_dilation, 1034
partial_sum_minimal_n, 1034
partition_chunk_share, 1034
partition_chunk_size, 1034
partition_minimal_n, 1034
qsb_steals, 1034
random_shuffle_minimal_n, 1034
replace_minimal_n, 1034
search_minimal_n, 1035
set, 1031
set_difference_minimal_n, 1035
set_intersection_minimal_n, 1035
set_symmetric_difference_minimal_n, 1035
set_union_minimal_n, 1035
sort_minimal_n, 1035
sort_mwms_oversampling, 1035
sort_qs_num_samples_preset, 1035
sort_qsb_base_case_maximal_n, 1035
TLB_size, 1036
transform_minimal_n, 1036
unique_copy_minimal_n, 1036
__gnu_parallel::_SplitConsistently< __exact, _RAIter, _Compare, _SortingPlacesIterator >, 1037
__gnu_parallel::_SplitConsistently< false, _RAIter, _Compare, _SortingPlacesIterator >, 1037
__gnu_parallel::_SplitConsistently< true, _RAIter, _Compare, _SortingPlacesIterator >, 1037
__gnu_parallel::__accumulate_binop_reduct< _BinOp >, 744
__gnu_parallel::__accumulate_selector< _It >, 744
_M_finish_iterator, 746
operator(), 745
__gnu_parallel::__adjacent_difference_selector< _It >, 746
_M_finish_iterator, 747
__gnu_parallel::__adjacent_find_selector, 747
_M_sequential_algorithm, 748
operator(), 748
__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 757
argument_type, 758
result_type, 758
__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 758
argument_type, 759
result_type, 759
__gnu_parallel::__count_if_selector< _It, _Diff >, 763
_M_finish_iterator, 765
operator(), 764
__gnu_parallel::__count_selector< _It, _Diff >, 765
_M_finish_iterator, 766
operator(), 766
__gnu_parallel::__fill_selector< _It >, 782
_M_finish_iterator, 783
operator(), 783
__gnu_parallel::__find_first_of_selector< _FIterator >, 783
_M_sequential_algorithm, 784
operator(), 784
__gnu_parallel::__find_if_selector, 785
_M_sequential_algorithm, 785
operator(), 786
__gnu_parallel::__for_each_selector< _It >, 786
_M_finish_iterator, 788

operator(), 787
__gnu_parallel::__generate_selector< _It >, 788
_M_finish_iterator, 789
operator(), 789
__gnu_parallel::__generic_find_selector, 789
__gnu_parallel::__generic_for_each_selector< _It >, 790
_M_finish_iterator, 792
__gnu_parallel::__identity_selector< _It >, 792
_M_finish_iterator, 793
operator(), 793
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 793
_M_finish_iterator, 795
__begin1_iterator, 795
__begin2_iterator, 795
__inner_product_selector, 794
operator(), 795
__gnu_parallel::__max_element_reduct< _Compare, _It >, 796
__gnu_parallel::__min_element_reduct< _Compare, _It >, 796
__gnu_parallel::__mismatch_selector, 798
_M_sequential_algorithm, 798
operator(), 799
__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 801
__gnu_parallel::__multiway_merge_3_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 802
__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< __sentinels, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 802
__gnu_parallel::__multiway_merge_4_variant_sentinel_switch< true, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 803
__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< __sentinels, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 803
__gnu_parallel::__multiway_merge_k_variant_sentinel_switch< false, __stable, _RAIterIterator, _RAIter3, _DifferenceTp, _Compare >, 803
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 819
_M_finish_iterator, 820
__new_val, 820
__replace_if_selector, 820
operator(), 820
__gnu_parallel::__replace_selector< _It, _Tp >, 821
_M_finish_iterator, 822
__new_val, 822
__replace_selector, 821
operator(), 822
__gnu_parallel::__transform1_selector< _It >, 823
_M_finish_iterator, 824
operator(), 823
__gnu_parallel::__transform2_selector< _It >, 824
_M_finish_iterator, 825
operator(), 825
__gnu_parallel::__unary_negate< _Predicate, argument_type >, 825
argument_type, 826
result_type, 826
__gnu_parallel::balanced_quicksort_tag, 1107
__get_num_threads, 1108
set_num_threads, 1108
__gnu_parallel::balanced_tag, 1109
__get_num_threads, 1109
set_num_threads, 1109
__gnu_parallel::constant_size_blocks_tag, 1919
__gnu_parallel::default_parallel_tag, 2049
__get_num_threads, 2050
set_num_threads, 2050
__gnu_parallel::equal_split_tag, 2135
__gnu_parallel::exact_tag, 2144
__get_num_threads, 2145
set_num_threads, 2145
__gnu_parallel::find_tag, 2162
__gnu_parallel::growing_blocks_tag, 2232
__gnu_parallel::multiway_mergesort_exact_tag, 2547
__get_num_threads, 2548
set_num_threads, 2548
__gnu_parallel::multiway_mergesort_sampling_tag, 2548
__get_num_threads, 2549
set_num_threads, 2549
__gnu_parallel::multiway_mergesort_tag, 2550
__get_num_threads, 2550
set_num_threads, 2550
__gnu_parallel::omp_loop_static_tag, 2685
__get_num_threads, 2685
set_num_threads, 2685
__gnu_parallel::omp_loop_tag, 2686
__get_num_threads, 2686
set_num_threads, 2686
__gnu_parallel::parallel_tag, 2724
__get_num_threads, 2726
parallel_tag, 2726
set_num_threads, 2726
__gnu_parallel::quicksort_tag, 2788
__get_num_threads, 2789
set_num_threads, 2789
__gnu_parallel::sampling_tag, 2869
__get_num_threads, 2869
set_num_threads, 2870
__gnu_parallel::sequential_tag, 2877
__gnu_parallel::unbalanced_tag, 3090
__get_num_threads, 3090
set_num_threads, 3090

__gnu_pbds, 473
__gnu_pbds::associative_tag, 1080
__gnu_pbds::basic_branch< Key, Mapped, Tag, Node_Update, Policy_Tl, _Alloc >, 1110
__gnu_pbds::basic_branch_tag, 1111
__gnu_pbds::basic_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Resize_Policy, Store_Hash, Tag, Policy_Tl, _Alloc >, 1183
__gnu_pbds::basic_hash_tag, 1185
__gnu_pbds::basic_invalidation_guarantee, 1226
__gnu_pbds::binary_heap_tag, 1787
__gnu_pbds::binomial_heap_tag, 1799
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1818
  cc_hash_max_collision_check_resize_trigger, 1820
  external_load_access, 1819
  get_load, 1820
  is_grow_needed, 1820
  is_resize_needed, 1820
  notify_cleared, 1820
  notify_erase_search_collision, 1820
  notify_erase_search_end, 1820
  notify_erase_search_start, 1820
  notify_erased, 1821
  notify_externally_resized, 1821
  notify_find_search_collision, 1821
  notify_find_search_end, 1821
  notify_find_search_start, 1821
  notify_insert_search_collision, 1821
  notify_insert_search_end, 1821
  notify_insert_search_start, 1822
  notify_inserted, 1822
  notify_resized, 1822
  set_load, 1822
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >, 1822
  cc_hash_table, 1824–1827
__gnu_pbds::cc_hash_tag, 1827
__gnu_pbds::container_error, 1930
  what, 1931
__gnu_pbds::container_tag, 1931
__gnu_pbds::container_traits< Cntnr >, 1932
  erase_can_throw, 1933
  order_preserving, 1933
  reverse_iteration, 1933
  split_join_can_throw, 1933
__gnu_pbds::container_traits_base< _Tag >, 1933
__gnu_pbds::container_traits_base< binary_heap_tag >, 1933
__gnu_pbds::container_traits_base< binomial_heap_tag >, 1934
__gnu_pbds::container_traits_base< cc_hash_tag >, 1934
__gnu_pbds::container_traits_base< gp_hash_tag >, 1934
__gnu_pbds::container_traits_base< list_update_tag >, 1935
__gnu_pbds::container_traits_base< ov_tree_tag >, 1935
__gnu_pbds::container_traits_base< pairing_heap_tag >, 1935
__gnu_pbds::container_traits_base< pat_trie_tag >, 1935
__gnu_pbds::container_traits_base< rb_tree_tag >, 1936
__gnu_pbds::container_traits_base< rc_binomial_heap_tag >, 1936
__gnu_pbds::container_traits_base< splay_tree_tag >, 1936
__gnu_pbds::container_traits_base< thin_heap_tag >, 1937
__gnu_pbds::detail::bin_search_tree_const_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >, 1764
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1766
  const_reference, 1767
  difference_type, 1767
  get_l_child, 1767
  get_metadata, 1768
  get_r_child, 1768
  iterator_category, 1767
  metadata_const_reference, 1767
  metadata_type, 1767
  operator!=, 1768
  operator==, 1768
  operator*, 1768
  reference, 1767
  value_type, 1767
__gnu_pbds::detail::bin_search_tree_it_< Node_Pointer, Value_Type, Pointer, Const_Pointer, Reference, Const_Reference, Is_Forward_Iterator, _Alloc >, 1768
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1770
  const_reference, 1771
  difference_type, 1771
  get_l_child, 1772
  get_metadata, 1772
  get_r_child, 1772
  iterator_category, 1771
  metadata_const_reference, 1771
  metadata_type, 1772
  operator!=, 1772
  operator==, 1773
  operator*, 1773
  reference, 1772

value_type, 1772
__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >, 1773
node_const_iterator, 1774
__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >, 1774
node_const_iterator, 1774
__gnu_pbds::detail::binary_heap< Value_Type, Cmp_Fn, _Alloc >, 1778
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1780
binary_heap_const_iterator_, 1782
const_pointer, 1781
const_reference, 1781
difference_type, 1781
iterator_category, 1781
operator!=, 1782
operator->, 1783
operator==, 1783
operator*, 1783
pointer, 1782
reference, 1782
value_type, 1782
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1783
binary_heap_point_const_iterator_, 1786
const_pointer, 1785
const_reference, 1785
difference_type, 1785
iterator_category, 1785
operator!=, 1786
operator->, 1786
operator==, 1786
operator*, 1786
pointer, 1785
reference, 1785
value_type, 1785
__gnu_pbds::detail::binomial_heap< Value_Type, Cmp_Fn, _Alloc >, 1794
__gnu_pbds::detail::binomial_heap_base< Value_Type, Cmp_Fn, _Alloc >, 1796
__gnu_pbds::detail::branch_policy< Node_CItr, Node_CItr, _Alloc >, 1815
__gnu_pbds::detail::branch_policy< Node_CItr, Node_Itr, _Alloc >, 1814
__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1828
empty, 1831
get_comb_hash_fn, 1831
get_eq_fn, 1832
get_hash_fn, 1832
get_resize_policy, 1832
__gnu_pbds::detail::cond_dealtor< Entry, _Alloc >, 1909
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, Tag, Policy_Tl >, 1921
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >, 1924
type, 1924
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >, 1924
type, 1925
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >, 1925
type, 1925
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >, 1925
type, 1926
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, pat_trie_tag, Policy_Tl >, 1926
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >, 1926
type, 1926
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >, 1927
type, 1927
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >, 1927
type, 1927
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >, 1928
type, 1928
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >, 1928
type, 1928
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >, 1929
type, 1929
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >, 1929
type, 1929
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, rb_tree_tag, Policy_Tl >, 1930
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >, 1930
type, 1930
__gnu_pbds::detail::container_base_dispatch< _VTp,

Cmp_Fn, _Alloc, binary_heap_tag, null_type >, 1921
type, 1922
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >, 1922
type, 1922
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >, 1922
type, 1923
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >, 1923
type, 1923
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag, null_type >, 1923
type, 1924
__gnu_pbds::detail::default_comb_hash_fn, 2046
type, 2046
__gnu_pbds::detail::default_eq_fn< Key >, 2048
type, 2048
__gnu_pbds::detail::default_hash_fn< Key >, 2049
type, 2049
__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >, 2050
type, 2050
__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >, 2051
type, 2051
__gnu_pbds::detail::default_trie_access_traits< Key >, 2051
__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >, 2052
type, 2052
__gnu_pbds::detail::default_update_policy, 2052
type, 2052
__gnu_pbds::detail::dumnode_const_iterator< Key, Data, _Alloc >, 2094
__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, No_Throw >, 2133
__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >, 2133
__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, false >::type, 3080
__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >, 2133
type, 2134
__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, No_Throw >, 2134
__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, false >, 2134
__gnu_pbds::detail::entry_pred< _VTp, Pred, _Alloc, true >, 2134
__gnu_pbds::detail::eq_by_less< Key, Cmp_Fn >, 2135
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2221
empty, 2224
get_comb_probe_fn, 2224
get_eq_fn, 2225
get_hash_fn, 2225
get_probe_fn, 2225
get_resize_policy, 2225, 2226
__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, Store_Hash >, 2251
__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, false >, 2251
__gnu_pbds::detail::hash_eq_fn< Key, Eq_Fn, _Alloc, true >, 2252
__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, Hold_Size >, 2256
__gnu_pbds::detail::hash_load_check_resize_trigger_size_base< Size_Type, true >, 2256
__gnu_pbds::detail::left_child_next_sibling_heap< Value_Type, Cmp_Fn, Node_Metadata, _Alloc >, 2334
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2336
const_pointer, 2337
const_reference, 2337
difference_type, 2337
iterator_category, 2337
left_child_next_sibling_heap_const_iterator_, 2337, 2338
operator!=, 2338
operator->, 2338
operator==, 2338
operator*, 2338
pointer, 2337
reference, 2337
value_type, 2337
__gnu_pbds::detail::left_child_next_sibling_heap_node_< _Value, _Metadata, _Alloc >, 2339
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator Node, _Alloc >, 2339
const_pointer, 2341
const_reference, 2341
difference_type, 2341
iterator_category, 2341
left_child_next_sibling_heap_node_point_const_iterator_, 2342
operator!=, 2342
operator->, 2342
operator==, 2342
operator*, 2342
pointer, 2341

reference, 2341
value_type, 2341
__gnu_pbds::detail::lu_counter_metadata< Size_Type >, 2401
__gnu_pbds::detail::lu_counter_policy_base< Size_Type >, 2403
__gnu_pbds::detail::lu_map< Key, Mapped, Eq_Fn, _Alloc, Update_Policy >, 2404
__gnu_pbds::detail::mask_based_range_hashing< Size_Type >, 2436
__gnu_pbds::detail::maybe_null_type< Key, Mapped, _Alloc, Store_Hash >, 2446
__gnu_pbds::detail::maybe_null_type< Key, null_type, _Alloc, Store_Hash >, 2448
__gnu_pbds::detail::mod_based_range_hashing< Size_Type >, 2467
__gnu_pbds::detail::no_throw_copies< Key, Mapped >, 2559
__gnu_pbds::detail::no_throw_copies< Key, null_type >, 2559
__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2698
node_begin, 2700
node_end, 2700
__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >::cond_dtor< Size_Type >, 1910
__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >, 2701
get_l_child, 2702
get_r_child, 2702
__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >, 2703
get_l_child, 2704
get_r_child, 2704
operator*, 2704
__gnu_pbds::detail::pairing_heap< Value_Type, Cmp_Fn, _Alloc >, 2722
__gnu_pbds::detail::pat_trie_base, 2737
node_type, 2738
__gnu_pbds::detail::pat_trie_base::_CIter< Node, Leaf, Head, Inode, Is_Forward_Iterator >, 886
__gnu_pbds::detail::pat_trie_base::_Head< _ATraits, Metadata >, 909
__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >, 910
__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::const_iterator, 1912
__gnu_pbds::detail::pat_trie_base::_Inode< _ATraits, Metadata >::iterator, 2328
__gnu_pbds::detail::pat_trie_base::_Iter< Node, Leaf, Head, Inode, Is_Forward_Iterator >, 912
__gnu_pbds::detail::pat_trie_base::_Leaf< _ATraits, Metadata >, 922
__gnu_pbds::detail::pat_trie_base::_Metadata< Metadata, _Alloc >, 951
__gnu_pbds::detail::pat_trie_base::_Metadata< null_type, _Alloc >, 951
__gnu_pbds::detail::pat_trie_base::_Node_base< _ATraits, Metadata >, 954
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 956
get_child, 957
get_metadata, 957
metadata_const_reference, 957
metadata_type, 957
num_children, 957
operator!=, 957
operator==, 958
operator*, 958
valid_prefix, 958
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 958
get_child, 960
get_metadata, 960
metadata_const_reference, 960
metadata_type, 960
num_children, 960
operator!=, 960
operator==, 961
operator*, 961
valid_prefix, 961
__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >, 2738
node_begin, 2740
node_end, 2741
node_type, 2740
__gnu_pbds::detail::probe_fn_base< _Alloc >, 2780
__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, Store_Hash >, 2794
__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, false >, 2794
__gnu_pbds::detail::ranged_hash_fn< Key, Hash_Fn, _Alloc, Comb_Hash_Fn, true >, 2795
__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, false >, 2795
__gnu_pbds::detail::ranged_hash_fn< Key, null_type, _Alloc, Comb_Hash_Fn, true >, 2796
__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, Store_Hash >, 2797
__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, false >, 2797
__gnu_pbds::detail::ranged_probe_fn< Key, Hash_Fn, _Alloc, Comb_Probe_Fn, Probe_Fn, true >,

2798
__gnu_pbds::detail::ranged_probe_fn< Key, null_type, _Alloc, Comb_Probe_Fn, null_type, false >, 2799
__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2810
node_begin, 2813
node_end, 2813
__gnu_pbds::detail::rb_tree_node_< Value_Type, Metadata, _Alloc >, 2814
__gnu_pbds::detail::rc< _Node, _Alloc >, 2815
__gnu_pbds::detail::rc_binomial_heap< Value_Type, Cmp_Fn, _Alloc >, 2816
__gnu_pbds::detail::rebind_traits< _Alloc, T >, 2819
__gnu_pbds::detail::resize_policy< _Tp >, 2843
__gnu_pbds::detail::select_value_type< Key, Mapped >, 2874
__gnu_pbds::detail::select_value_type< Key, null_type >, 2874
__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2928
node_begin, 2931
node_end, 2931
__gnu_pbds::detail::splay_tree_node_< Value_Type, Metadata, _Alloc >, 2931
__gnu_pbds::detail::stored_data< _Tv, _Th, Store_Hash >, 2975
__gnu_pbds::detail::stored_data< _Tv, _Th, false >, 2976
__gnu_pbds::detail::stored_hash< _Th >, 2977
__gnu_pbds::detail::stored_value< _Tv >, 2977
__gnu_pbds::detail::synth_access_traits< Type_Traits, Set, _ATraits >, 2992
__gnu_pbds::detail::thin_heap< Value_Type, Cmp_Fn, _Alloc >, 2997
__gnu_pbds::detail::tree_metadata_helper< Node_Update, _BTp >, 3045
__gnu_pbds::detail::tree_metadata_helper< Node_Update, false >, 3045
__gnu_pbds::detail::tree_metadata_helper< Node_Update, true >, 3045
__gnu_pbds::detail::tree_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >, 3046
__gnu_pbds::detail::tree_traits< Key, Data, Cmp_Fn, Node_Update, Tag, _Alloc >, 3048
__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, 3049
node_const_iterator, 3049
__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, 3049
node_const_iterator, 3051
__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, 3051
node_const_iterator, 3052
__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, 3052
node_const_iterator, 3053
__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, 3053
node_const_iterator, 3054
__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, 3054
node_const_iterator, 3055
__gnu_pbds::detail::trie_metadata_helper< Node_Update, _BTp >, 3058
__gnu_pbds::detail::trie_metadata_helper< Node_Update, false >, 3058
__gnu_pbds::detail::trie_metadata_helper< Node_Update, true >, 3058
__gnu_pbds::detail::trie_node_metadata_dispatch< Key, Data, Cmp_Fn, Node_Update, _Alloc >, 3059
__gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3062
end, 3063
__gnu_pbds::detail::trie_traits< Key, Data, _ATraits, Node_Update, Tag, _Alloc >, 3069
__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3070
node_const_iterator, 3070
node_update, 3070
synth_access_traits, 3070
__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3071
node_const_iterator, 3071
node_update, 3071
synth_access_traits, 3072
__gnu_pbds::detail::types_traits< Key, Mapped, _Alloc, Store_Hash >, 3084
__gnu_pbds::direct_mask_range_hashing< Size_Type >, 2078
operator(), 2079
__gnu_pbds::direct_mod_range_hashing< Size_Type >, 2079
operator(), 2080
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >, 2215
gp_hash_table, 2216–2220
__gnu_pbds::gp_hash_tag, 2220
__gnu_pbds::hash_exponential_size_policy< Size_Type >, 2252
hash_exponential_size_policy, 2253
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2253
external_load_access, 2255
get_loads, 2255
hash_load_check_resize_trigger, 2255
notify_cleared, 2255
notify_inserted, 2255

notify_resized, 2255
set_loads, 2256
__gnu_pbds::hash_prime_size_policy, 2262
hash_prime_size_policy, 2262
size_type, 2262
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2264
external_load_access, 2265
get_actual_size, 2266
get_loads, 2266
get_new_size, 2266
get_size_policy, 2267
get_trigger_policy, 2267
hash_standard_resize_policy, 2266
resize, 2267
set_loads, 2267
__gnu_pbds::insert_error, 2280
what, 2281
__gnu_pbds::join_error, 2333
what, 2334
__gnu_pbds::linear_probe_fn< Size_Type >, 2354
operator(), 2355
__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >, 2379
list_update, 2380
__gnu_pbds::list_update_tag, 2380
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2401
max_count, 2403
metadata_reference, 2402
metadata_type, 2402
operator(), 2403
__gnu_pbds::lu_move_to_front_policy< _Alloc >, 2407
metadata_reference, 2407
metadata_type, 2407
operator(), 2407
__gnu_pbds::null_node_update< _Tp1, _Tp2, _Tp3, _Tp4 >, 2566
__gnu_pbds::null_type, 2567
__gnu_pbds::ov_tree_tag, 2704
__gnu_pbds::pairing_heap_tag, 2724
__gnu_pbds::pat_trie_tag, 2741
__gnu_pbds::point_invalidation_guarantee, 2761
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >, 2773
priority_queue, 2774
__gnu_pbds::priority_queue_tag, 2780
__gnu_pbds::quadratic_probe_fn< Size_Type >, 2785
operator(), 2785
__gnu_pbds::range_invalidation_guarantee, 2793
__gnu_pbds::rb_tree_tag, 2814
__gnu_pbds::rc_binomial_heap_tag, 2818
__gnu_pbds::resize_error, 2842
what, 2843
__gnu_pbds::sample_probe_fn, 2855
operator(), 2856
sample_probe_fn, 2856
swap, 2856
__gnu_pbds::sample_range_hashing, 2856
notify_resized, 2857
operator(), 2857
sample_range_hashing, 2857
size_type, 2857
swap, 2857
__gnu_pbds::sample_ranged_hash_fn, 2857
notify_resized, 2858
operator(), 2858
sample_ranged_hash_fn, 2858
swap, 2858
__gnu_pbds::sample_ranged_probe_fn, 2858
__gnu_pbds::sample_resize_policy, 2859
get_new_size, 2860
is_resize_needed, 2860
notify_cleared, 2860
notify_erase_search_collision, 2860
notify_erase_search_end, 2860
notify_erase_search_start, 2860
notify_erased, 2860
notify_find_search_collision, 2860
notify_find_search_end, 2861
notify_find_search_start, 2861
notify_insert_search_collision, 2861
notify_insert_search_end, 2861
notify_insert_search_start, 2861
notify_inserted, 2861
notify_resized, 2861
sample_range_hashing, 2861
sample_resize_policy, 2860
size_type, 2860
swap, 2861
__gnu_pbds::sample_resize_trigger, 2862
is_grow_needed, 2863
is_resize_needed, 2863
notify_cleared, 2863
notify_erase_search_collision, 2863
notify_erase_search_end, 2863
notify_erase_search_start, 2863
notify_erased, 2863
notify_externally_resized, 2863
notify_find_search_collision, 2863
notify_find_search_end, 2863
notify_find_search_start, 2863
notify_insert_search_collision, 2864
notify_insert_search_end, 2864
notify_insert_search_start, 2864
notify_inserted, 2864
notify_resized, 2864

sample_range_hashing, 2864
sample_resize_trigger, 2862
size_type, 2862
swap, 2864
__gnu_pbds::sample_size_policy, 2864
get_nearest_larger_size, 2865
get_nearest_smaller_size, 2865
sample_range_hashing, 2865
sample_size_policy, 2865
size_type, 2865
swap, 2865
__gnu_pbds::sample_tree_node_update< Const_Node_Iter, Node_Iter, Cmp_Fn, _Alloc >, 2866
__gnu_pbds::sample_trie_access_traits, 2866
begin, 2866
e_pos, 2866
e_type, 2866
end, 2867
__gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 2867
operator(), 2867
sample_trie_node_update, 2867
__gnu_pbds::sample_update_policy, 2868
metadata_type, 2868
operator(), 2868
sample_update_policy, 2868
swap, 2869
__gnu_pbds::sequence_tag, 2877
__gnu_pbds::splay_tree_tag, 2932
__gnu_pbds::string_tag, 2978
__gnu_pbds::thin_heap_tag, 2999
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >, 3042
cmp_fn, 3044
tree, 3044
__gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >, 3046
find_by_order, 3047
operator(), 3047
order_of_key, 3047
__gnu_pbds::tree_tag, 3048
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >, 3055
access_traits, 3057
trie, 3057
__gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3059
find_by_order, 3061
operator(), 3061
order_of_key, 3061
order_of_prefix, 3061
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3063
a_const_iterator, 3065
access_traits, 3065
allocator_type, 3065
operator(), 3065
prefix_range, 3065, 3066
size_type, 3065
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3066
begin, 3067
const_iterator, 3067
e_pos, 3068
e_type, 3067
end, 3068
__gnu_pbds::trie_tag, 3068
__gnu_pbds::trivial_iterator_tag, 3072
__gnu_sequential, 475
__init_winner
__gnu_parallel::_LoserTree< false, _Tp, _Compare >, 938
__inner_product_selector
__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 794
__insert_start
__gnu_parallel::_LoserTree< __stable, _Tp, _Compare >, 936
__gnu_parallel::_LoserTree< false, _Tp, _Compare >, 938
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 940
__int_traits
__gnu_cxx, 403
__invoke
Utilities, 305
__is_sorted
__gnu_parallel, 441
__lg
std, 590
__median
SGI, 155
__median_of_three_iterators
__gnu_parallel, 442
__merge_advance
__gnu_parallel, 442
__merge_advance_movc
__gnu_parallel, 443
__merge_advance_usual
__gnu_parallel, 444
__multiline
std::regex_constants, 719
__new_val
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 820
__gnu_parallel::__replace_selector< _It, _Tp >, 822
__num_bitmaps
__gnu_cxx::__detail, 414

__num_blocks
__gnu_cxx::__detail, 414
__num_get_type
std::basic_ios< _CharT, _Traits >, 1231
std::basic_ofstream< _CharT, _Traits >, 1383
std::basic_ostream< _CharT, _Traits >, 1416
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1449
__num_put_type
std::basic_ifstream< _CharT, _Traits >, 1191
std::basic_ios< _CharT, _Traits >, 1231
std::basic_istream< _CharT, _Traits >, 1303
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1343
__parallel_merge_advance
__gnu_parallel, 444, 445
__parallel_nth_element
__gnu_parallel, 445
__parallel_partial_sort
__gnu_parallel, 446
__parallel_partial_sum
__gnu_parallel, 446
__parallel_partial_sum_basecase
__gnu_parallel, 447
__parallel_partial_sum_linear
__gnu_parallel, 447
__parallel_partition
__gnu_parallel, 448
__parallel_random_shuffle
__gnu_parallel, 448
__parallel_random_shuffle_drs
__gnu_parallel, 448
__parallel_random_shuffle_drs_pu
__gnu_parallel, 449
__parallel_sort
__gnu_parallel, 449–454
__parallel_sort_qs
__gnu_parallel, 454
__parallel_sort_qs_conquer
__gnu_parallel, 455
__parallel_sort_qs_divide
__gnu_parallel, 455
__parallel_sort_qsb
__gnu_parallel, 456
__parallel_unique_copy
__gnu_parallel, 456, 457
__polynomial
std::regex_constants, 719
__ptr_rebind
std, 587
__qsb_conquer
__gnu_parallel, 457
__qsb_divide
__gnu_parallel, 458
__qsb_local_sort_with_helping
__gnu_parallel, 458
__random_number_pow2
__gnu_parallel, 458
__rd_log2
__gnu_parallel, 459
__replace_if_selector
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 820
__replace_selector
__gnu_parallel::__replace_selector< _It, _Tp >, 821
__resize_and_overwrite
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1511
std::basic_string< _CharT, _Traits, _Alloc >, 1598
__round_up_to_pow2
__gnu_parallel, 459
__search_template
__gnu_parallel, 459
__sequential_multiway_merge
__gnu_parallel, 460
__sequential_random_shuffle
__gnu_parallel, 460
__shrink
__gnu_parallel, 461
__shrink_and_double
__gnu_parallel, 461
__static_pointer_cast
__gnu_cxx, 403
__streambuf_type
std::basic_streambuf< _CharT, _Traits >, 1489
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3252
__umap_traits
std, 587
__ummap_traits
std, 587
__umset_traits
std, 587
__uset_traits
std, 587
__valid_range
__gnu_debug, 422
__verbose_terminate_handler
Exceptions, 135
__versa_string
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 830–833
__yield
__gnu_parallel, 462
~_LoserTreeBase
__gnu_parallel::_LoserTreeBase< _Tp, _Compare >, 940
~_RestrictedBoundedConcurrentQueue

__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >, 977
~_Safe_sequence_base
__gnu_debug::_Safe_sequence_base, 1018
~_Safe_unordered_container_base
__gnu_debug::_Safe_unordered_container_base, 1024
~__versa_string
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 833
~any
std::experimental::fundamentals_v1::any, 1077
~auto_ptr
std::auto_ptr< _Tp >, 1094
~basic_filebuf
std::basic_filebuf< _CharT, _Traits >, 1115
~basic_fstream
std::basic_fstream< _CharT, _Traits >, 1143
~basic_ifstream
std::basic_ifstream< _CharT, _Traits >, 1194
~basic_ios
std::basic_ios< _CharT, _Traits >, 1234
~basic_iostream
std::basic_iostream< _CharT, _Traits >, 1259
~basic_istream
std::basic_istream< _CharT, _Traits >, 1306
~basic_istringstream
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1346
~basic_ofstream
std::basic_ofstream< _CharT, _Traits >, 1386
~basic_ostream
std::basic_ostream< _CharT, _Traits >, 1419
~basic_ostringstream
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1452
~basic_regex
std::basic_regex< _Ch_type, _Rx_traits >, 1481
~basic_streambuf
std::basic_streambuf< _CharT, _Traits >, 1490
~basic_string
std::basic_string< _CharT, _Traits, _Alloc >, 1594, 1598
~basic_stringstream
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1721
~collate
std::__cxx11::collate< _CharT >, 1896
~ctype
std::ctype< char >, 1961
std::ctype< wchar_t >, 1990
~deque
std::deque< _Tp, _Alloc >, 2062
~facet
std::locale::facet, 2157
~forward_list
std::forward_list< _Tp, _Alloc >, 2175
~gslice
Numeric Arrays, 225
~ios_base
std::ios_base, 2292
~list
std::list< _Tp, _Alloc >, 2364
~locale
std::locale, 2385
~map
std::map< _Key, _Tp, _Compare, _Alloc >, 2418
~match_results
std::match_results< _Bi_iter, _Alloc >, 2440
~messages
std::messages< _CharT >, 2461
~money_get
std::money_get< _CharT, _InIter >, 2472
~money_put
std::money_put< _CharT, _OutIter >, 2477
~moneypunct
std::moneypunct< _CharT, _Intl >, 2483
~multimap
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2512
~multiset
std::multiset< _Key, _Compare, _Alloc >, 2536
~num_get
std::num_get< _CharT, _InIter >, 2570
~num_put
std::num_put< _CharT, _OutIter >, 2587
~numpunct
std::numpunct< _CharT >, 2678
~sentry
std::basic_ostream< _CharT, _Traits >::sentry, 2876
~set
std::set< _Key, _Compare, _Alloc >, 2887
~stdio_filebuf
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2940
~temporary_buffer
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 2996
~time_get
std::time_get< _CharT, _InIter >, 3011
~time_put
std::time_put< _CharT, _OutIter >, 3034
~type_info
__gnu_debug::type_info, 3082
std::type_info, 3083
~unique_ptr
std::unique_ptr< _Tp, _Dp >, 3101
std::unique_ptr< _Tp[], _Dp >, 3107
~vector

std::vector< _Tp, _Alloc >, 3232

a
std::extreme_value_distribution< _RealType >, 2153
std::weibull_distribution< _RealType >, 3270
a_const_iterator
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3065
abi, 475
abs
Complex Numbers, 185
math.h, 3625
std, 591
stdlib.h, 3685
Time, 375
access_traits
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >, 3057
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3065
accumulate
Generalized Numeric operations, 6, 7
accumulate_minimal_n
__gnu_parallel::_Settings, 1031
acosh
std, 591
Adaptors for pointers to functions, 317
ptr_fun, 317
Adaptors for pointers to members, 318
add_lvalue_reference_t
Metaprogramming, 359
add_pointer_t
Metaprogramming, 359
add_rvalue_reference_t
Metaprogramming, 359
addressof
Utilities, 305
adjacent_difference
Generalized Numeric operations, 7, 8
adjacent_difference_minimal_n
__gnu_parallel::_Settings, 1031
adjacent_find
Non-Mutating, 45
adjustfield
std::basic_fstream< _CharT, _Traits >, 1178
std::basic_ifstream< _CharT, _Traits >, 1222
std::basic_ios< _CharT, _Traits >, 1246
std::basic_iostream< _CharT, _Traits >, 1293
std::basic_istream< _CharT, _Traits >, 1332
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1373
std::basic_ofstream< _CharT, _Traits >, 1407
std::basic_ostream< _CharT, _Traits >, 1438
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1471
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1756
std::ios_base, 2297
adopt_lock
Mutexes, 124
advance
std, 591
airy_ai
Mathematical Special Functions, 195
airy_aif
Mathematical Special Functions, 195
airy_ail
Mathematical Special Functions, 195
airy_bi
Mathematical Special Functions, 195
airy_bif
Mathematical Special Functions, 195
airy_bil
Mathematical Special Functions, 196
algo.h, 3640
algobase.h, 3650
algorithm, 3275, 3277
algorithmfwd.h, 3286, 3291
Algorithms, 4
align
Memory, 326
align.h, 3299
aligned_buffer.h, 3533
aligned_storage_t
Metaprogramming, 359
alignment_value
Metaprogramming, 362
all
std::bitset< _Nb >, 1807
std::locale, 2388
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2101
all_of
Non-Mutating, 45
alloc_traits.h, 3300
allocate
__gnu_cxx::__alloc_traits< _Alloc, typename >, 751, 752
std::allocator_traits< _Alloc >, 1057
std::allocator_traits< allocator< _Tp > >, 1062
std::allocator_traits< allocator< void > >, 1067
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1071
allocate_at_least
__gnu_cxx::__alloc_traits< _Alloc, typename >, 752
std::allocator_traits< _Alloc >, 1058
std::allocator_traits< allocator< _Tp > >, 1063

std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1072
allocate_shared
Pointer Abstractions, 337
allocated_ptr.h, 3300
allocator.h, 3300
allocator_type
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3065
std::allocator_traits< _Alloc >, 1055
std::allocator_traits< allocator< _Tp > >, 1061
std::allocator_traits< allocator< void > >, 1066
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1070
std::set< _Key, _Compare, _Alloc >, 2883
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3116
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3145
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3170
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3194
Allocators, 332
__allocator_base, 333
operator==, 333
alpha
std::gamma_distribution< _RealType >, 2210
any, 3277
std::bitset< _Nb >, 1807
std::experimental::fundamentals_v1::any, 1076, 1077
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2101
any_cast
Type-safe container of any type, 294–296
any_of
Non-Mutating, 46
app
std::basic_fstream< _CharT, _Traits >, 1179
std::basic_ifstream< _CharT, _Traits >, 1222
std::basic_ios< _CharT, _Traits >, 1246
std::basic_iostream< _CharT, _Traits >, 1293
std::basic_istream< _CharT, _Traits >, 1333
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1373
std::basic_ofstream< _CharT, _Traits >, 1407
std::basic_ostream< _CharT, _Traits >, 1438
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1471
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1756
std::ios_base, 2297
append
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 833–836
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1511–1513
std::basic_string< _CharT, _Traits, _Alloc >, 1598–1603
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2101
apply
Numeric Arrays, 225, 226
apply_generator
__gnu_cxx::typelist, 415
arg
Complex Numbers, 185
std::tr1, 729
argument_type
__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >, 1775
__gnu_cxx::select1st< _Pair >, 2873
__gnu_cxx::select2nd< _Pair >, 2873
__gnu_cxx::subtractive_rng, 2992
__gnu_cxx::unary_compose< _Operation1, _Operation2 >, 3087
__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 758
__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 759
__gnu_parallel::__unary_negate< _Predicate, argument_type >, 826
std::binder1st< _Operation >, 1789
std::binder2nd< _Operation >, 1791
std::const_mem_fun_ref_t< _Ret, _Tp >, 1917
std::const_mem_fun_t< _Ret, _Tp >, 1918
std::hash< __gnu_cxx::throw_value_limit >, 2239
std::hash< __gnu_cxx::throw_value_random >, 2240
std::logical_not< _Tp >, 2394
std::logical_not< void >, 2395
std::mem_fun_ref_t< _Ret, _Tp >, 2451
std::mem_fun_t< _Ret, _Tp >, 2452
std::negate< _Tp >, 2552
std::negate< void >, 2553
std::pointer_to_unary_function< _Arg, _Result >, 2764
std::unary_function< _Arg, _Result >, 3088
std::unary_negate< _Predicate >, 3089
Arithmetic Function Object Classes, 319
array, 3278, 3279
Array creation functions, 287
make_array, 287
to_array, 287
assertions.h, 3503
assign
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 836–839

__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1514, 1515
std::basic_regex< _Ch_type, _Rx_traits >, 1481–1483
std::basic_string< _CharT, _Traits, _Alloc >, 1604–1610
std::deque< _Tp, _Alloc >, 2066
std::error_condition, 2144
std::forward_list< _Tp, _Alloc >, 2175, 2176
std::list< _Tp, _Alloc >, 2364, 2365
std::vector< _Tp, _Alloc >, 3232, 3233
assoc_container.hpp, 3538
assoc_laguerre
Mathematical Special Functions, 196
TR1 Mathematical Special Functions, 252
assoc_laguerref
Mathematical Special Functions, 197
assoc_laguerrel
Mathematical Special Functions, 197
assoc_legendre
Mathematical Special Functions, 197
TR1 Mathematical Special Functions, 252
assoc_legendref
Mathematical Special Functions, 198
assoc_legendrel
Mathematical Special Functions, 198
Associative, 126
assume_aligned
Memory, 327
async
Futures, 120
at
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 840
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1516–1518
std::basic_string< _CharT, _Traits, _Alloc >, 1610–1612
std::deque< _Tp, _Alloc >, 2067
std::map< _Key, _Tp, _Compare, _Alloc >, 2418
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3120–3122
std::vector< _Tp, _Alloc >, 3233, 3234
ate
std::basic_fstream< _CharT, _Traits >, 1179
std::basic_ifstream< _CharT, _Traits >, 1222
std::basic_ios< _CharT, _Traits >, 1246
std::basic_iostream< _CharT, _Traits >, 1293
std::basic_istream< _CharT, _Traits >, 1333
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1373
std::basic_ofstream< _CharT, _Traits >, 1407
std::basic_ostream< _CharT, _Traits >, 1438
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1472
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1756
std::ios_base, 2297
atomic, 3279
atomic_base.h, 3301
atomic_bool
Atomics, 109
ATOMIC_BOOL_LOCK_FREE
Atomics, 109
atomic_char
Atomics, 109
atomic_char16_t
Atomics, 109
atomic_char32_t
Atomics, 109
atomic_compare_exchange_strong
Pointer Abstractions, 338
atomic_compare_exchange_strong_explicit
Pointer Abstractions, 339
atomic_compare_exchange_weak
Pointer Abstractions, 340
atomic_compare_exchange_weak_explicit
Pointer Abstractions, 341
atomic_exchange
Pointer Abstractions, 342
atomic_exchange_explicit
Pointer Abstractions, 342, 343
atomic_futex.h, 3302
atomic_int
Atomics, 110
atomic_int16_t
Atomics, 110
atomic_int32_t
Atomics, 110
atomic_int64_t
Atomics, 110
atomic_int8_t
Atomics, 110
atomic_int_fast16_t
Atomics, 110
atomic_int_fast32_t
Atomics, 110
atomic_int_fast64_t
Atomics, 111
atomic_int_fast8_t
Atomics, 111
atomic_int_least16_t
Atomics, 111
atomic_int_least32_t
Atomics, 111
atomic_int_least64_t
Atomics, 111

atomic_int_least8_t
Atomics, 111
atomic_intmax_t
Atomics, 111
atomic_intptr_t
Atomics, 112
atomic_is_lock_free
Pointer Abstractions, 343, 344
atomic_llong
Atomics, 112
atomic_load
Pointer Abstractions, 344
atomic_load_explicit
Pointer Abstractions, 345
atomic_lockfree_defines.h, 3302
atomic_long
Atomics, 112
atomic_ptrdiff_t
Atomics, 112
atomic_schar
Atomics, 112
atomic_short
Atomics, 112
atomic_size_t
Atomics, 112
atomic_store
Pointer Abstractions, 345, 346
atomic_store_explicit
Pointer Abstractions, 346, 347
atomic_timed_wait.h, 3302
atomic_uchar
Atomics, 113
atomic_uint
Atomics, 113
atomic_uint16_t
Atomics, 113
atomic_uint32_t
Atomics, 113
atomic_uint64_t
Atomics, 113
atomic_uint8_t
Atomics, 113
atomic_uint_fast16_t
Atomics, 113
atomic_uint_fast32_t
Atomics, 114
atomic_uint_fast64_t
Atomics, 114
atomic_uint_fast8_t
Atomics, 114
atomic_uint_least16_t
Atomics, 114
atomic_uint_least32_t
Atomics, 114
atomic_uint_least64_t
Atomics, 114
atomic_uint_least8_t
Atomics, 114
atomic_uintmax_t
Atomics, 115
atomic_uintptr_t
Atomics, 115
atomic_ullong
Atomics, 115
atomic_ulong
Atomics, 115
atomic_ushort
Atomics, 115
atomic_wait.h, 3302
atomic_wchar_t
Atomics, 115
atomic_word.h, 3714
atomicity.h, 3533
Atomics, 104
atomic_bool, 109
ATOMIC_BOOL_LOCK_FREE, 109
atomic_char, 109
atomic_char16_t, 109
atomic_char32_t, 109
atomic_int, 110
atomic_int16_t, 110
atomic_int32_t, 110
atomic_int64_t, 110
atomic_int8_t, 110
atomic_int_fast16_t, 110
atomic_int_fast32_t, 110
atomic_int_fast64_t, 111
atomic_int_fast8_t, 111
atomic_int_least16_t, 111
atomic_int_least32_t, 111
atomic_int_least64_t, 111
atomic_int_least8_t, 111
atomic_intmax_t, 111
atomic_intptr_t, 112
atomic_llong, 112
atomic_long, 112
atomic_ptrdiff_t, 112
atomic_schar, 112
atomic_short, 112
atomic_size_t, 112
atomic_uchar, 113
atomic_uint, 113
atomic_uint16_t, 113
atomic_uint32_t, 113
atomic_uint64_t, 113
atomic_uint8_t, 113
atomic_uint_fast16_t, 113
atomic_uint_fast32_t, 114

atomic_uint_fast64_t, 114
atomic_uint_fast8_t, 114
atomic_uint_least16_t, 114
atomic_uint_least32_t, 114
atomic_uint_least64_t, 114
atomic_uint_least8_t, 114
atomic_uintmax_t, 115
atomic_uintptr_t, 115
atomic_ullong, 115
atomic_ulong, 115
atomic_ushort, 115
atomic_wchar_t, 115
kill_dependency, 116
memory_order, 116
auto_ptr
std::auto_ptr< _Tp >, 1094, 1095
auto_ptr.h, 3283
awk
std::regex_constants, 720

b
std::extreme_value_distribution< _RealType >, 2153
std::weibull_distribution< _RealType >, 3270
back
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 840, 841
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1519
std::basic_string< _CharT, _Traits, _Alloc >, 1612
std::deque< _Tp, _Alloc >, 2068
std::list< _Tp, _Alloc >, 2365
std::queue< _Tp, _Sequence >, 2787
std::vector< _Tp, _Alloc >, 3234
back_insert_iterator
std::back_insert_iterator< _Container >, 1099
back_inserter
Iterators, 175
backward_warning.h, 3283
bad
std::basic_fstream< _CharT, _Traits >, 1143
std::basic_ifstream< _CharT, _Traits >, 1194
std::basic_ios< _CharT, _Traits >, 1235
std::basic_iostream< _CharT, _Traits >, 1259
std::basic_istream< _CharT, _Traits >, 1306
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1347
std::basic_ofstream< _CharT, _Traits >, 1386
std::basic_ostream< _CharT, _Traits >, 1419
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1452
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1722
badbit
std::basic_fstream< _CharT, _Traits >, 1179
std::basic_ifstream< _CharT, _Traits >, 1222
std::basic_ios< _CharT, _Traits >, 1247
std::basic_iostream< _CharT, _Traits >, 1294
std::basic_istream< _CharT, _Traits >, 1333
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1374
std::basic_ofstream< _CharT, _Traits >, 1407
std::basic_ostream< _CharT, _Traits >, 1439
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1472
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1756
std::ios_base, 2297
balanced_quicksort.h, 3652
barrier, 3285
base
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 989
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1003
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2085
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2273
std::reverse_iterator< _Iterator >, 2847
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2918
Base and Implementation Classes, 277
_Opcode, 278
Base and Policy Classes, 145–147
base.h, 3652
basefield
std::basic_fstream< _CharT, _Traits >, 1179
std::basic_ifstream< _CharT, _Traits >, 1222
std::basic_ios< _CharT, _Traits >, 1247
std::basic_iostream< _CharT, _Traits >, 1294
std::basic_istream< _CharT, _Traits >, 1333
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1374
std::basic_ofstream< _CharT, _Traits >, 1407
std::basic_ostream< _CharT, _Traits >, 1439
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1472
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1756
std::ios_base, 2297
basic
std::regex_constants, 720
basic_file.h, 3714
basic_filebuf
std::basic_filebuf< _CharT, _Traits >, 1115
basic_fstream
std::basic_fstream< _CharT, _Traits >, 1142, 1143
basic_ifstream

std::basic_ifstream< _CharT, _Traits >, 1193, 1194
basic_ios
std::basic_ios< _CharT, _Traits >, 1234, 1235
basic_ios.h, 3302
basic_ios.tcc, 3303
basic_iostream
std::basic_iostream< _CharT, _Traits >, 1259
basic_istream
std::basic_istream< _CharT, _Traits >, 1306
basic_istringstream
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1346
basic_iterator.h, 3653
basic_ofstream
std::basic_ofstream< _CharT, _Traits >, 1385, 1386
basic_ostream
std::basic_ostream< _CharT, _Traits >, 1419
basic_ostringstream
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1451
basic_regex
std::basic_regex< _Ch_type, _Rx_traits >, 1478–1480
basic_streambuf
std::basic_streambuf< _CharT, _Traits >, 1490
basic_string
std::basic_string< _CharT, _Traits, _Alloc >, 1591–1597
basic_string.h, 3303
basic_string.tcc, 3306
basic_stringbuf
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1696, 1697
basic_stringstream
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1721
before
__gnu_debug::type_info, 3082
std::type_info, 3083
before_begin
std::forward_list< _Tp, _Alloc >, 2176
beg
std::basic_fstream< _CharT, _Traits >, 1179
std::basic_ifstream< _CharT, _Traits >, 1222
std::basic_ios< _CharT, _Traits >, 1247
std::basic_iostream< _CharT, _Traits >, 1294
std::basic_istream< _CharT, _Traits >, 1333
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1374
std::basic_ofstream< _CharT, _Traits >, 1407
std::basic_ostream< _CharT, _Traits >, 1439
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1472
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1756
std::ios_base, 2297
begin
__gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >, 1040
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 841
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 2997
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >, 970
__gnu_pbds::sample_trie_access_traits, 2866
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3067
Filesystem, 166
std, 592
std::_Temporary_buffer< _ForwardIterator, _Tp >, 1042
std::basic_string< _CharT, _Traits, _Alloc >, 1613
std::deque< _Tp, _Alloc >, 2068
std::forward_list< _Tp, _Alloc >, 2177
std::list< _Tp, _Alloc >, 2365, 2366
std::map< _Key, _Tp, _Compare, _Alloc >, 2418, 2419
std::match_results< _Bi_iter, _Alloc >, 2441
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2512
std::multiset< _Key, _Compare, _Alloc >, 2536
std::set< _Key, _Compare, _Alloc >, 2888
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3122, 3123
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3150
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3175
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3199
std::valarray< _Tp >, 3219
std::vector< _Tp, _Alloc >, 3234
Bernoulli Distributions, 238
operator<<, 238, 239
operator>>, 239
bernoulli_distribution
std::bernoulli_distribution, 1761
beta
Mathematical Special Functions, 198
std::gamma_distribution< _RealType >, 2210
TR1 Mathematical Special Functions, 252
betaf
Mathematical Special Functions, 199
betal
Mathematical Special Functions, 199

bin_search_tree_.hpp, 3539
binary
    std::basic_fstream< _CharT, _Traits >, 1179
    std::basic_ifstream< _CharT, _Traits >, 1223
    std::basic_ios< _CharT, _Traits >, 1247
    std::basic_iostream< _CharT, _Traits >, 1294
    std::basic_istream< _CharT, _Traits >, 1333
    std::basic_istringstream< _CharT, _Traits, _Alloc >, 1374
    std::basic_ofstream< _CharT, _Traits >, 1407
    std::basic_ostream< _CharT, _Traits >, 1439
    std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1472
    std::basic_stringstream< _CharT, _Traits, _Alloc >, 1756
    std::ios_base, 2298
Binary Search, 85
    binary_search, 86
    equal_range, 86, 87
    lower_bound, 87, 88
    upper_bound, 88, 89
binary_heap_.hpp, 3542
binary_heap_const_iterator_
    __gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1782
binary_heap_point_const_iterator_
    __gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1786
binary_search
    Binary Search, 86
bind
    Binder Classes, 321
bind1st
    Binder Classes, 321
bind2nd
    Binder Classes, 321
Binder Classes, 319
    bind, 321
    bind1st, 321
    bind2nd, 321
binders.h, 3306, 3307
binomial_heap_.hpp, 3556
binomial_heap_base_.hpp, 3556
bit, 3286
Bit manipulation, 181
bitmap_allocator.h, 3533
    _BALLOC_ALIGN_BYTES, 3534
bitset, 3470, 3471
    std::bitset< _Nb >, 1805, 1806
bool_set, 3694
    std::tr2::bool_set, 1814
bool_set.tcc, 3695
boolalpha
    std, 593
    std::basic_fstream< _CharT, _Traits >, 1179
    std::basic_ifstream< _CharT, _Traits >, 1223
    std::basic_ios< _CharT, _Traits >, 1247
    std::basic_iostream< _CharT, _Traits >, 1294
    std::basic_istream< _CharT, _Traits >, 1333
    std::basic_istringstream< _CharT, _Traits, _Alloc >, 1374
    std::basic_ofstream< _CharT, _Traits >, 1407
    std::basic_ostream< _CharT, _Traits >, 1439
    std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1472
    std::basic_stringstream< _CharT, _Traits, _Alloc >, 1757
    std::ios_base, 2298
Boolean Operations Classes, 322
boost_concept_check.h, 3307
Branch-Based, 144
branch_policy.hpp, 3557
bucket
    __gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1003
bucket_count
    std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3123
    std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3150
    std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3176
    std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3200

c
    std::queue< _Tp, _Sequence >, 2788
c++0x_warning.h, 3308
c++allocator.h, 3714
c++config.h, 3715
c++io.h, 3721
c++locale.h, 3721
c++locale_internal.h, 3721
c_str
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 841
    std::basic_string< _CharT, _Traits, _Alloc >, 1613
cache_line_size
    __gnu_parallel::_Settings, 1031
call_once
    Mutexes, 122
    std::once_flag, 2687
capacity
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 841
    __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1519

std::basic_string< _CharT, _Traits, _Alloc >, 1613, 1614
std::vector< _Tp, _Alloc >, 3235
cassert, 3471
cast.h, 3534
category
std::error_code, 2142
std::error_condition, 2144
std::locale, 2382
cbefore_begin
std::forward_list< _Tp, _Alloc >, 2177
cbegin
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 841
std, 593
std::basic_string< _CharT, _Traits, _Alloc >, 1614
std::deque< _Tp, _Alloc >, 2068
std::forward_list< _Tp, _Alloc >, 2177
std::list< _Tp, _Alloc >, 2366
std::map< _Key, _Tp, _Compare, _Alloc >, 2419
std::match_results< _Bi_iter, _Alloc >, 2441
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2512
std::multiset< _Key, _Compare, _Alloc >, 2536
std::set< _Key, _Compare, _Alloc >, 2888
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3123
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3151
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3176
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3200
std::vector< _Tp, _Alloc >, 3235
cc_hash_max_collision_check_resize_trigger
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1820
cc_hash_max_collision_check_resize_trigger_imp.hpp, 3575
cc_hash_table
__gnu_pbds::cc_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Hash_Fn, Resize_Policy, Store_Hash, _Alloc >, 1824–1827
cc_ht_map_.hpp, 3557
ccomplex, 3471, 3472
cctype, 3472
ceil
chrono.h, 3312
Time, 375, 376
cend
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 842
std, 593
std::basic_string< _CharT, _Traits, _Alloc >, 1614
std::deque< _Tp, _Alloc >, 2068
std::forward_list< _Tp, _Alloc >, 2177
std::list< _Tp, _Alloc >, 2366
std::map< _Key, _Tp, _Compare, _Alloc >, 2419
std::match_results< _Bi_iter, _Alloc >, 2441
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2512
std::multiset< _Key, _Compare, _Alloc >, 2536
std::set< _Key, _Compare, _Alloc >, 2888
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3124
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3151
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3176
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3200
std::vector< _Tp, _Alloc >, 3235
cerr
std, 649
cerrno, 3472
cfenv, 3472, 3473
cfloat, 3473
char_traits.h, 3308
char_type
std::__ctype_abstract_base< _CharT >, 768
std::__cxx11::collate< _CharT >, 1895
std::__cxx11::collate_byname< _CharT >, 1900
std::basic_ios< _CharT, _Traits >, 1232
std::basic_streambuf< _CharT, _Traits >, 1489
std::ctype< char >, 1960
std::ctype< wchar_t >, 1990
std::istreambuf_iterator< _CharT, _Traits >, 2326
std::messages< _CharT >, 2460
std::money_get< _CharT, _InIter >, 2472
std::money_put< _CharT, _OutIter >, 2476
std::moneypunct< _CharT, _Intl >, 2482
std::num_get< _CharT, _InIter >, 2570
std::num_put< _CharT, _OutIter >, 2587
std::numpunct< _CharT >, 2677
std::ostream_iterator< _Tp, _CharT, _Traits >, 2690
std::ostreambuf_iterator< _CharT, _Traits >, 2694
std::time_get< _CharT, _InIter >, 3011
std::time_put< _CharT, _OutIter >, 3034
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3252
charconv, 3473
charconv.h, 3308
chars_format
std, 589
checkers.h, 3653
chrono, 3475, 3477
chrono.h, 3309
ceil, 3312
chrono_io.h, 3313

cin
std, 649
cinttypes, 3477, 3478
ciso646, 3478
clamp
Sorting, 65
classic
std::locale, 2385
classic_table
std::ctype< char >, 1961
std::ctype_byname< char >, 2019
clear
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 842
std::basic_fstream< _CharT, _Traits >, 1144
std::basic_ifstream< _CharT, _Traits >, 1195
std::basic_ios< _CharT, _Traits >, 1235
std::basic_iostream< _CharT, _Traits >, 1260
std::basic_istream< _CharT, _Traits >, 1306
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1347
std::basic_ofstream< _CharT, _Traits >, 1387
std::basic_ostream< _CharT, _Traits >, 1419
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1452
std::basic_string< _CharT, _Traits, _Alloc >, 1614
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1722
std::deque< _Tp, _Alloc >, 2068
std::error_condition, 2144
std::experimental::fundamentals_v1::any, 1077
std::forward_list< _Tp, _Alloc >, 2177
std::list< _Tp, _Alloc >, 2366
std::map< _Key, _Tp, _Compare, _Alloc >, 2419
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2513
std::multiset< _Key, _Compare, _Alloc >, 2536
std::set< _Key, _Compare, _Alloc >, 2888
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2101
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3124
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3152
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3177
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3201
std::vector< _Tp, _Alloc >, 3235
climits, 3478
clocale, 3478
clock_cast
Time, 376
clog
std, 649
close
__gnu_cxx::enc_filebuf< _CharT >, 2115
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2940
std::basic_filebuf< _CharT, _Traits >, 1116
std::basic_fstream< _CharT, _Traits >, 1144
std::basic_ifstream< _CharT, _Traits >, 1195
std::basic_ofstream< _CharT, _Traits >, 1387
cmath, 3479, 3485, 3486
cmp_fn
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >, 3044
cmp_fn_imps.hpp, 3558
code
std::regex_error, 2827
codecvt, 3488
codecvt.h, 3316
codecvt_specializations.h, 3535
collate
std::__cxx11::collate< _CharT >, 1895, 1896
std::locale, 2388
std::regex_constants, 720
combine
std::locale, 2385
common_reference_t
Metaprogramming, 359
common_type_t
Metaprogramming, 359
comp_ellint_1
Mathematical Special Functions, 199
TR1 Mathematical Special Functions, 252
comp_ellint_1f
Mathematical Special Functions, 200
comp_ellint_1l
Mathematical Special Functions, 200
comp_ellint_2
Mathematical Special Functions, 200
TR1 Mathematical Special Functions, 253
comp_ellint_2f
Mathematical Special Functions, 201
comp_ellint_2l
Mathematical Special Functions, 201
comp_ellint_3
Mathematical Special Functions, 201
TR1 Mathematical Special Functions, 253
comp_ellint_3f
Mathematical Special Functions, 202
comp_ellint_3l
Mathematical Special Functions, 202
compare, 3725
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 842–844
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1519–1525
std::__cxx11::collate< _CharT >, 1896

std::basic_string< _CharT, _Traits, _Alloc >, 1614–1620
std::sub_match< _BiIter >, 2984
compare_three_way_result_t
std, 587
Comparison Classes, 322
compatibility.h, 3722
compiletime_settings.h, 3653
_GLIBCXX_CALL, 3654
_GLIBCXX_PARALLEL_ASSERTIONS, 3654
_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_L1, 3654
_GLIBCXX_RANDOM_SHUFFLE_CONSIDER_TLB, 3654
_GLIBCXX_SCALE_DOWN_FPU, 3654
_GLIBCXX_VERBOSE_LEVEL, 3654
complex, 3489, 3493
std::complex< _Tp >, 1904
Complex Numbers, 181
abs, 185
arg, 185
conj, 185
cos, 185
cosh, 185
exp, 185
fabs, 185
log, 185
log10, 186
norm, 186
operator<<, 188
operator>>, 188
operator+, 186, 187
operator-, 187
operator/, 187, 188
operator==, 188
operator*, 186
polar, 189
pow, 189
sin, 189
sinh, 190
sqrt, 190
tan, 190
tanh, 190
complex.h, 3494
compose1
SGI, 157
compose2
SGI, 157
concept_check.h, 3317
Concepts, 314
concurrence.h, 3536
Concurrency, 116
cond_dealtor.hpp, 3562
cond_key_dtor_entry_dealtor.hpp, 3558
Condition Variables, 117
cv_status, 117
condition_variable, 3496
conditional_t
Metaprogramming, 359
conf_hyperg
Mathematical Special Functions, 203
std::tr1, 729
conf_hypergf
Mathematical Special Functions, 203
std::tr1, 729
conf_hypergl
Mathematical Special Functions, 203
std::tr1, 730
conj
Complex Numbers, 185
Const-propagating wrapper, 288
const_iterator
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3067
std::set< _Key, _Compare, _Alloc >, 2883
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3116
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3145
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3170
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3194
const_iterator.hpp, 3543
const_local_iterator
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3116
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3145
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3171
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3195
const_pointer
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1781
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1785
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2337
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_ite
Node, _Alloc >, 2341
std::allocator_traits< _Alloc >, 1055
std::allocator_traits< allocator< _Tp > >, 1061
std::allocator_traits< allocator< void > >, 1066
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1070

std::set< _Key, _Compare, _Alloc >, 2883
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3116
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3145
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3171
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3195
const_pointer_cast
Pointer Abstractions, 347
std, 593
const_reference
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1767
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1771
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1781
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1785
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2337
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >, 2341
std::set< _Key, _Compare, _Alloc >, 2883
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3117
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3146
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3171
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3195
const_reverse_iterator
std::set< _Key, _Compare, _Alloc >, 2883
const_void_pointer
__gnu_cxx::__alloc_traits< _Alloc, typename >, 750
std::allocator_traits< _Alloc >, 1055
std::allocator_traits< allocator< _Tp > >, 1061
std::allocator_traits< allocator< void > >, 1066
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1070
constant0
SGI, 157
constant1
SGI, 158
constant2
SGI, 158
construct
__gnu_cxx::__alloc_traits< _Alloc, typename >, 753
std::allocator_traits< _Alloc >, 1058
std::allocator_traits< allocator< _Tp > >, 1063
std::allocator_traits< allocator< void > >, 1067
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1072
constructor_destructor_fn_imps.hpp, 3559
constructor_destructor_no_store_hash_fn_imps.hpp, 3559
constructor_destructor_store_hash_fn_imps.hpp, 3559
constructors_destructor_fn_imps.hpp, 3544, 3545
container_base_dispatch.hpp, 3562
container_type
std::back_insert_iterator< _Container >, 1099
std::front_insert_iterator< _Container >, 2193
std::insert_iterator< _Container >, 2283
Containers, 125, 144
contains
std::map< _Key, _Tp, _Compare, _Alloc >, 2419
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2513
std::multiset< _Key, _Compare, _Alloc >, 2537
std::set< _Key, _Compare, _Alloc >, 2888
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3124, 3125
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3152
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3177
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3201
converted
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3273
copy
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 845
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1526
Mutating, 23
std::basic_string< _CharT, _Traits, _Alloc >, 1620, 1621
copy_backward
Mutating, 24
copy_if
Mutating, 24
copy_n
Mutating, 25
SGI, 158
copy_options
Filesystem, 166
Filesystem TS, 285
copyable_function
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1939, 1940
copyfmt
std::basic_fstream< _CharT, _Traits >, 1144
std::basic_ifstream< _CharT, _Traits >, 1195

std::basic_ios< _CharT, _Traits >, 1236
std::basic_iostream< _CharT, _Traits >, 1260
std::basic_istream< _CharT, _Traits >, 1307
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1347
std::basic_ofstream< _CharT, _Traits >, 1387
std::basic_ostream< _CharT, _Traits >, 1420
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1453
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1722
cos
Complex Numbers, 185
math.h, 3625
cosh
Complex Numbers, 185
math.h, 3625
count
Non-Mutating, 46
std::bitset< _Nb >, 1807
std::map< _Key, _Tp, _Compare, _Alloc >, 2420
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2513
std::multiset< _Key, _Compare, _Alloc >, 2537
std::set< _Key, _Compare, _Alloc >, 2889
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2101
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3125
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3152
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3178
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3202
count_if
Non-Mutating, 47
count_minimal_n
__gnu_parallel::_Settings, 1031
cout
std, 649
cow_string.h, 3317
cpp_type_traits.h, 3318
cpu_defines.h, 3722
cpyfunc_impl.h, 3318
crbegin
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 846
std, 594
std::basic_string< _CharT, _Traits, _Alloc >, 1621, 1622
std::deque< _Tp, _Alloc >, 2069
std::list< _Tp, _Alloc >, 2366
std::map< _Key, _Tp, _Compare, _Alloc >, 2420
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2514
std::multiset< _Key, _Compare, _Alloc >, 2538
std::set< _Key, _Compare, _Alloc >, 2889
std::vector< _Tp, _Alloc >, 3235
cref
std::reference_wrapper< _Tp >, 2825
cregex_token_iterator
Regular Expressions, 259
crend
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 846
std, 594
std::basic_string< _CharT, _Traits, _Alloc >, 1622
std::deque< _Tp, _Alloc >, 2069
std::list< _Tp, _Alloc >, 2366
std::map< _Key, _Tp, _Compare, _Alloc >, 2420
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2514
std::multiset< _Key, _Compare, _Alloc >, 2538
std::set< _Key, _Compare, _Alloc >, 2889
std::vector< _Tp, _Alloc >, 3235
csetjmp, 3497
cshift
Numeric Arrays, 226
csignal, 3497
cstdalign, 3497
cstdarg, 3498
cstdbool, 3498
cstddef, 3499
cstdint, 3499
cstdio, 3499, 3500
cstdlib, 3500
cstring, 3500
csub_match
Regular Expressions, 259
ctgmath, 3501
ctime, 3501, 3502
ctype
std::ctype< char >, 1961
std::ctype< wchar_t >, 1990
std::locale, 2388
ctype_base.h, 3722
ctype_inline.h, 3723
cuchar, 3502
cur
std::basic_fstream< _CharT, _Traits >, 1180
std::basic_ifstream< _CharT, _Traits >, 1223
std::basic_ios< _CharT, _Traits >, 1247
std::basic_iostream< _CharT, _Traits >, 1294
std::basic_istream< _CharT, _Traits >, 1334
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1374
std::basic_ofstream< _CharT, _Traits >, 1408
std::basic_ostream< _CharT, _Traits >, 1439

std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1472
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1757
std::ios_base, 2298
curr_symbol
std::moneypunct< _CharT, _Intl >, 2483
std::moneypunct_byname< _CharT, _Intl >, 2491
current_exception
Exceptions, 135
cv_status
Condition Variables, 117
cwchar, 3502, 3503
cwctype, 3503
cxxabi.h, 3726
__cxa_demangle, 3727
cxxabi_forced.h, 3318
cxxabi_init_exception.h, 3319
cxxabi_tweaks.h, 3723
cyl_bessel_i
Mathematical Special Functions, 203
TR1 Mathematical Special Functions, 253
cyl_bessel_if
Mathematical Special Functions, 204
cyl_bessel_il
Mathematical Special Functions, 204
cyl_bessel_j
Mathematical Special Functions, 204
TR1 Mathematical Special Functions, 253
cyl_bessel_jf
Mathematical Special Functions, 205
cyl_bessel_jl
Mathematical Special Functions, 205
cyl_bessel_k
Mathematical Special Functions, 205
TR1 Mathematical Special Functions, 253
cyl_bessel_kf
Mathematical Special Functions, 206
cyl_bessel_kl
Mathematical Special Functions, 206
cyl_neumann
Mathematical Special Functions, 207
TR1 Mathematical Special Functions, 253
cyl_neumannf
Mathematical Special Functions, 207
cyl_neumannl
Mathematical Special Functions, 207

data
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 846
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1526
std, 594, 595
std::basic_string< _CharT, _Traits, _Alloc >, 1622, 1623
std::vector< _Tp, _Alloc >, 3236
Data parallel extensions, 300
Data Structure Type, 150
date_order
std::time_get< _CharT, _InIter >, 3011
std::time_get_byname< _CharT, _InIter >, 3023
days
Time, 374
deallocate
__gnu_cxx::__alloc_traits< _Alloc, typename >, 753, 754
std::allocator_traits< _Alloc >, 1058
std::allocator_traits< allocator< _Tp > >, 1064
std::allocator_traits< allocator< void > >, 1068
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1073
debug.h, 3504
debug_allocator.h, 3536
debug_fn_imps.hpp, 3545, 3546
debug_map_base.hpp, 3563
debug_no_store_hash_fn_imps.hpp, 3559
debug_store_hash_fn_imps.hpp, 3560
debugging, 3519
dec
std, 595
std::basic_fstream< _CharT, _Traits >, 1180
std::basic_ifstream< _CharT, _Traits >, 1223
std::basic_ios< _CharT, _Traits >, 1247
std::basic_iostream< _CharT, _Traits >, 1294
std::basic_istream< _CharT, _Traits >, 1334
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1374
std::basic_ofstream< _CharT, _Traits >, 1408
std::basic_ostream< _CharT, _Traits >, 1439
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1473
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1757
std::ios_base, 2298
decay_t
Metaprogramming, 359
decimal, 3519
Decimal Floating-Point Arithmetic, 190
decimal128
std::decimal::decimal128, 2042
decimal32
std::decimal::decimal32, 2044
decimal32_to_long_long
std::decimal, 695
decimal64
std::decimal::decimal64, 2045
decimal_point

std::moneypunct< _CharT, _Intl >, 2483
std::moneypunct_byname< _CharT, _Intl >, 2491
std::numpunct< _CharT >, 2678
std::numpunct_byname< _CharT >, 2682
declval
Utilities, 306
default_delete
std::default_delete< _Tp >, 2047
std::default_delete< _Tp[]>, 2048
default_error_condition
std::error_category, 2140
std::error_code, 2142
default_sentinel
std, 649
defaultfloat
std, 595
defer_lock
Mutexes, 124
denorm_absent
std, 589
denorm_indeterminate
std, 589
denorm_min
std::numeric_limits< _Tp >, 2599
denorm_present
std, 589
densities
std::piecewise_constant_distribution< _RealType >, 2754
std::piecewise_linear_distribution< _RealType >, 2758
Deprecated List, 1
deque, 3528, 3529
std::deque< _Tp, _Alloc >, 2060–2062
deque.tcc, 3319
destroy
__gnu_cxx::__alloc_traits< _Alloc, typename >, 754
std::allocator_traits< _Alloc >, 1059
std::allocator_traits< allocator< _Tp > >, 1064
std::allocator_traits< allocator< void > >, 1068
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1073
destroying_delete
std, 649
detected_or
Detection idiom, 290
detected_or_t
Detection idiom, 290
detected_t
Detection idiom, 290
Detection idiom, 289
__cpp_lib_experimental_detect, 290
detected_or, 290
detected_or_t, 290
detected_t, 290
is_detected, 290
is_detected_convertible, 291
is_detected_convertible_v, 291
is_detected_exact, 291
is_detected_exact_v, 291
is_detected_v, 291
void_t, 291
Diagnostics, 129
generic_category, 130
make_error_code, 130
make_error_condition, 130
operator<<, 130
operator<=>, 131
operator==, 131, 132
system_category, 132
difference_type
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1767
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1771
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1781
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1785
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2337
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_ite
Node, _Alloc >, 2341
std::allocator_traits< _Alloc >, 1056
std::allocator_traits< allocator< _Tp > >, 1061
std::allocator_traits< allocator< void > >, 1066
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1070
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2323
std::istreambuf_iterator< _CharT, _Traits >, 2326
std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, 2331
std::ostream_iterator< _Tp, _CharT, _Traits >, 2690
std::ostreambuf_iterator< _CharT, _Traits >, 2694
std::pointer_traits< _Ptr >, 2765
std::pointer_traits< _Tp * >, 2766
std::raw_storage_iterator< _OutputIterator, _Tp >, 2806
std::set< _Key, _Compare, _Alloc >, 2883
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3117
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3146
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3171
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3195

digits
std::__numeric_limits_base, 807
std::numeric_limits< _Tp >, 2601
std::numeric_limits< bool >, 2605
std::numeric_limits< char >, 2609
std::numeric_limits< char16_t >, 2613
std::numeric_limits< char32_t >, 2617
std::numeric_limits< double >, 2621
std::numeric_limits< float >, 2625
std::numeric_limits< int >, 2629
std::numeric_limits< long >, 2633
std::numeric_limits< long double >, 2637
std::numeric_limits< long long >, 2641
std::numeric_limits< short >, 2645
std::numeric_limits< signed char >, 2649
std::numeric_limits< unsigned char >, 2653
std::numeric_limits< unsigned int >, 2657
std::numeric_limits< unsigned long >, 2661
std::numeric_limits< unsigned long long >, 2665
std::numeric_limits< unsigned short >, 2669
std::numeric_limits< wchar_t >, 2673
digits10
std::__numeric_limits_base, 807
std::numeric_limits< _Tp >, 2601
std::numeric_limits< bool >, 2605
std::numeric_limits< char >, 2609
std::numeric_limits< char16_t >, 2613
std::numeric_limits< char32_t >, 2617
std::numeric_limits< double >, 2621
std::numeric_limits< float >, 2625
std::numeric_limits< int >, 2629
std::numeric_limits< long >, 2633
std::numeric_limits< long double >, 2637
std::numeric_limits< long long >, 2641
std::numeric_limits< short >, 2645
std::numeric_limits< signed char >, 2649
std::numeric_limits< unsigned char >, 2653
std::numeric_limits< unsigned int >, 2657
std::numeric_limits< unsigned long >, 2661
std::numeric_limits< unsigned long long >, 2665
std::numeric_limits< unsigned short >, 2669
std::numeric_limits< wchar_t >, 2673
direct_mask_range_hashing_imp.hpp, 3564
direct_mod_range_hashing_imp.hpp, 3564
directory_options
Filesystem, 166
Filesystem TS, 285
discard
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2085
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2273
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2351
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2456
std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2750
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2918
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2989
discard_block_engine
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2084, 2085
distance
SGI, 158
std, 595
do_allocate
std::pmr::monotonic_buffer_resource, 2498
std::pmr::unsynchronized_pool_resource, 3215
do_always_noconv
std::codecvt< _InternT, _ExternT, _StateT >, 1846
std::codecvt< _InternT, _ExternT, encoding_state >, 1852
std::codecvt< char, char, mbstate_t >, 1859
std::codecvt< char16_t, char, mbstate_t >, 1867
std::codecvt< char32_t, char, mbstate_t >, 1874
std::codecvt< wchar_t, char, mbstate_t >, 1882
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1890
do_compare
std::__cxx11::collate< _CharT >, 1897
do_curr_symbol
std::moneypunct< _CharT, _Intl >, 2484
std::moneypunct_byname< _CharT, _Intl >, 2491
do_date_order
std::time_get< _CharT, _InIter >, 3012
std::time_get_byname< _CharT, _InIter >, 3023
do_deallocate
std::pmr::monotonic_buffer_resource, 2498
std::pmr::unsynchronized_pool_resource, 3215
do_decimal_point
std::moneypunct< _CharT, _Intl >, 2484
std::moneypunct_byname< _CharT, _Intl >, 2491
std::numpunct< _CharT >, 2678
std::numpunct_byname< _CharT >, 2682
do_encoding
std::codecvt< _InternT, _ExternT, _StateT >, 1846
std::codecvt< _InternT, _ExternT, encoding_state >, 1852
std::codecvt< char, char, mbstate_t >, 1859
std::codecvt< char16_t, char, mbstate_t >, 1867
std::codecvt< char32_t, char, mbstate_t >, 1874
std::codecvt< wchar_t, char, mbstate_t >, 1882
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1890

do_falsename
std::numpunct< _CharT >, 2678
std::numpunct_byname< _CharT >, 2682
do_frac_digits
std::moneypunct< _CharT, _Intl >, 2484
std::moneypunct_byname< _CharT, _Intl >, 2492
do_get
std::messages< _CharT >, 2461
std::messages_byname< _CharT >, 2464
std::money_get< _CharT, _InIter >, 2473
std::num_get< _CharT, _InIter >, 2570–2577
std::time_get< _CharT, _InIter >, 3012
std::time_get_byname< _CharT, _InIter >, 3023
do_get_date
std::time_get< _CharT, _InIter >, 3013
std::time_get_byname< _CharT, _InIter >, 3024
do_get_monthname
std::time_get< _CharT, _InIter >, 3013
std::time_get_byname< _CharT, _InIter >, 3024
do_get_time
std::time_get< _CharT, _InIter >, 3014
std::time_get_byname< _CharT, _InIter >, 3025
do_get_weekday
std::time_get< _CharT, _InIter >, 3015
std::time_get_byname< _CharT, _InIter >, 3026
do_get_year
std::time_get< _CharT, _InIter >, 3015
std::time_get_byname< _CharT, _InIter >, 3026
do_grouping
std::moneypunct< _CharT, _Intl >, 2484
std::moneypunct_byname< _CharT, _Intl >, 2492
std::numpunct< _CharT >, 2678
std::numpunct_byname< _CharT >, 2683
do_hash
std::__cxx11::collate< _CharT >, 1897
do_in
std::codecvt< _InternT, _ExternT, _StateT >, 1847
std::codecvt< _InternT, _ExternT, encoding_state >, 1852
std::codecvt< char, char, mbstate_t >, 1860
std::codecvt< char16_t, char, mbstate_t >, 1867
std::codecvt< char32_t, char, mbstate_t >, 1874
std::codecvt< wchar_t, char, mbstate_t >, 1882
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1890
do_is
std::__ctype_abstract_base< _CharT >, 769
std::ctype< _CharT >, 1946
std::ctype< char >, 1961, 1962
std::ctype< wchar_t >, 1991
std::ctype_byname< _CharT >, 2005
std::ctype_byname< char >, 2019, 2020
do_is_equal
std::pmr::monotonic_buffer_resource, 2498
std::pmr::unsynchronized_pool_resource, 3215
do_length
std::codecvt< _InternT, _ExternT, _StateT >, 1847
std::codecvt< _InternT, _ExternT, encoding_state >, 1852
std::codecvt< char, char, mbstate_t >, 1860
std::codecvt< char16_t, char, mbstate_t >, 1867
std::codecvt< char32_t, char, mbstate_t >, 1875
std::codecvt< wchar_t, char, mbstate_t >, 1882
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1891
do_max_length
std::codecvt< _InternT, _ExternT, _StateT >, 1847
std::codecvt< _InternT, _ExternT, encoding_state >, 1852
std::codecvt< char, char, mbstate_t >, 1860
std::codecvt< char16_t, char, mbstate_t >, 1867
std::codecvt< char32_t, char, mbstate_t >, 1875
std::codecvt< wchar_t, char, mbstate_t >, 1882
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1891
do_narrow
std::__ctype_abstract_base< _CharT >, 769, 770
std::ctype< _CharT >, 1946, 1947
std::ctype< char >, 1963, 1964
std::ctype< wchar_t >, 1991, 1992
std::ctype_byname< _CharT >, 2005, 2006
std::ctype_byname< char >, 2021, 2022
do_neg_format
std::moneypunct< _CharT, _Intl >, 2485
std::moneypunct_byname< _CharT, _Intl >, 2492
do_negative_sign
std::moneypunct< _CharT, _Intl >, 2485
std::moneypunct_byname< _CharT, _Intl >, 2493
do_out
std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 760
std::codecvt< _InternT, _ExternT, _StateT >, 1847
std::codecvt< _InternT, _ExternT, encoding_state >, 1853
std::codecvt< char, char, mbstate_t >, 1860
std::codecvt< char16_t, char, mbstate_t >, 1867
std::codecvt< char32_t, char, mbstate_t >, 1875
std::codecvt< wchar_t, char, mbstate_t >, 1882
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1891
do_pos_format
std::moneypunct< _CharT, _Intl >, 2485
std::moneypunct_byname< _CharT, _Intl >, 2493
do_positive_sign
std::moneypunct< _CharT, _Intl >, 2486
std::moneypunct_byname< _CharT, _Intl >, 2493
do_put
std::money_put< _CharT, _OutIter >, 2477, 2478

std::num_put< _CharT, _OutIter >, 2588–2591
std::time_put< _CharT, _OutIter >, 3035
std::time_put_byname< _CharT, _OutIter >, 3038
do_scan_is
std::__ctype_abstract_base< _CharT >, 771
std::ctype< _CharT >, 1948
std::ctype< char >, 1965
std::ctype< wchar_t >, 1992
std::ctype_byname< _CharT >, 2007
std::ctype_byname< char >, 2023
do_scan_not
std::__ctype_abstract_base< _CharT >, 771
std::ctype< _CharT >, 1948
std::ctype< char >, 1966
std::ctype< wchar_t >, 1993
std::ctype_byname< _CharT >, 2007
std::ctype_byname< char >, 2023, 2024
do_thousands_sep
std::moneypunct< _CharT, _Intl >, 2486
std::moneypunct_byname< _CharT, _Intl >, 2494
std::numpunct< _CharT >, 2679
std::numpunct_byname< _CharT >, 2683
do_tolower
std::__ctype_abstract_base< _CharT >, 772
std::ctype< _CharT >, 1949
std::ctype< char >, 1967, 1968
std::ctype< wchar_t >, 1993, 1994
std::ctype_byname< _CharT >, 2008
std::ctype_byname< char >, 2024–2026
do_toupper
std::__ctype_abstract_base< _CharT >, 773
std::ctype< _CharT >, 1949, 1950
std::ctype< char >, 1969, 1970
std::ctype< wchar_t >, 1994, 1995
std::ctype_byname< _CharT >, 2008, 2009
std::ctype_byname< char >, 2026, 2027
do_transform
std::__cxx11::collate< _CharT >, 1898
do_truename
std::numpunct< _CharT >, 2679
std::numpunct_byname< _CharT >, 2683
do_unshift
std::codecvt< _InternT, _ExternT, _StateT >, 1847
std::codecvt< _InternT, _ExternT, encoding_state >, 1853
std::codecvt< char, char, mbstate_t >, 1860
std::codecvt< char16_t, char, mbstate_t >, 1868
std::codecvt< char32_t, char, mbstate_t >, 1875
std::codecvt< wchar_t, char, mbstate_t >, 1883
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1891
do_widen
std::__ctype_abstract_base< _CharT >, 773, 774
std::ctype< _CharT >, 1950, 1951
std::ctype< char >, 1970, 1971
std::ctype< wchar_t >, 1995
std::ctype_byname< _CharT >, 2009, 2010
std::ctype_byname< char >, 2028, 2029
duration_cast
Time, 377
Dynamic Bitset., 139
operator!=, 140
operator<<, 141
operator<=, 141
operator>, 141
operator>>, 142
operator>=, 141
operator-, 140
operator&, 140
operator^, 142
operator|, 142
dynamic_bitset, 3695
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2099–2101
dynamic_bitset.tcc, 3696
dynamic_pointer_cast
Pointer Abstractions, 347
std, 596

e_pos
__gnu_pbds::sample_trie_access_traits, 2866
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3068
e_type
__gnu_pbds::sample_trie_access_traits, 2866
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3067
eback
__gnu_cxx::enc_filebuf< _CharT >, 2115
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2941
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2961
std::basic_filebuf< _CharT, _Traits >, 1116
std::basic_streambuf< _CharT, _Traits >, 1490
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1697
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3254
ECMAScript
std::regex_constants, 720
egptr
__gnu_cxx::enc_filebuf< _CharT >, 2115
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2941
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2961
std::basic_filebuf< _CharT, _Traits >, 1116
std::basic_streambuf< _CharT, _Traits >, 1491
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1697

std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3254
egrep
std::regex_constants, 720
element_type
std::auto_ptr< _Tp >, 1093
std::pointer_traits< _Ptr >, 2765
std::pointer_traits< _Tp * >, 2766
std::shared_ptr< _Tp >, 2908
ellint_1
Mathematical Special Functions, 208
TR1 Mathematical Special Functions, 253
ellint_1f
Mathematical Special Functions, 208
ellint_1l
Mathematical Special Functions, 208
ellint_2
Mathematical Special Functions, 209
TR1 Mathematical Special Functions, 254
ellint_2f
Mathematical Special Functions, 209
ellint_2l
Mathematical Special Functions, 210
ellint_3
Mathematical Special Functions, 210
TR1 Mathematical Special Functions, 254
ellint_3f
Mathematical Special Functions, 211
ellint_3l
Mathematical Special Functions, 211
emplace
std::deque< _Tp, _Alloc >, 2069
std::list< _Tp, _Alloc >, 2366
std::map< _Key, _Tp, _Compare, _Alloc >, 2420
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2514
std::multiset< _Key, _Compare, _Alloc >, 2538
std::set< _Key, _Compare, _Alloc >, 2889
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3125
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3153
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3178
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3202
std::vector< _Tp, _Alloc >, 3236
emplace_after
std::forward_list< _Tp, _Alloc >, 2178
emplace_front
std::forward_list< _Tp, _Alloc >, 2178
emplace_hint
std::map< _Key, _Tp, _Compare, _Alloc >, 2421
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2515
std::multiset< _Key, _Compare, _Alloc >, 2538
std::set< _Key, _Compare, _Alloc >, 2890
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3126
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3153
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3178
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3202
empty
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 846
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1526
__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1831
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2224
std, 596
std::basic_string< _CharT, _Traits, _Alloc >, 1623
std::deque< _Tp, _Alloc >, 2069
std::experimental::fundamentals_v1::any, 1077
std::forward_list< _Tp, _Alloc >, 2178
std::list< _Tp, _Alloc >, 2367
std::map< _Key, _Tp, _Compare, _Alloc >, 2421
std::match_results< _Bi_iter, _Alloc >, 2441
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2515
std::multiset< _Key, _Compare, _Alloc >, 2539
std::priority_queue< _Tp, _Sequence, _Compare >, 2779
std::queue< _Tp, _Sequence >, 2787
std::set< _Key, _Compare, _Alloc >, 2890
std::stack< _Tp, _Sequence >, 2935
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2102
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3126
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3154
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3179
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3203
std::vector< _Tp, _Alloc >, 3236
enable_if_t
Metaprogramming, 359
enable_special_members.h, 3321
enable_view
ranges_base.h, 3378
enc_filebuf.h, 3536

end
__gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >, 1040
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 846
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 2997
__gnu_parallel::_PseudoSequence< _Tp, _DifferenceTp >, 970
__gnu_pbds::detail::trie_policy_base< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3063
__gnu_pbds::sample_trie_access_traits, 2867
__gnu_pbds::trie_string_access_traits< String, Min_E_Val, Max_E_Val, Reverse, _Alloc >, 3068
Filesystem, 166, 167
std, 597
std::_Temporary_buffer< _ForwardIterator, _Tp >, 1042
std::basic_fstream< _CharT, _Traits >, 1180
std::basic_ifstream< _CharT, _Traits >, 1223
std::basic_ios< _CharT, _Traits >, 1248
std::basic_iostream< _CharT, _Traits >, 1295
std::basic_istream< _CharT, _Traits >, 1334
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1375
std::basic_ofstream< _CharT, _Traits >, 1408
std::basic_ostream< _CharT, _Traits >, 1440
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1473
std::basic_string< _CharT, _Traits, _Alloc >, 1623
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1757
std::deque< _Tp, _Alloc >, 2069, 2070
std::forward_list< _Tp, _Alloc >, 2178, 2179
std::ios_base, 2298
std::list< _Tp, _Alloc >, 2367
std::map< _Key, _Tp, _Compare, _Alloc >, 2422
std::match_results< _Bi_iter, _Alloc >, 2441
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2515
std::multiset< _Key, _Compare, _Alloc >, 2539
std::set< _Key, _Compare, _Alloc >, 2890
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3127
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3154
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3179, 3180
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3203, 3204
std::valarray< _Tp >, 3219
std::vector< _Tp, _Alloc >, 3236
endl
std, 598
ends
std, 598
entry_cmp.hpp, 3546
entry_list_fn_imps.hpp, 3560
entry_metadata_base.hpp, 3569
entry_pred.hpp, 3547
eof
std::basic_fstream< _CharT, _Traits >, 1144
std::basic_ifstream< _CharT, _Traits >, 1195
std::basic_ios< _CharT, _Traits >, 1236
std::basic_iostream< _CharT, _Traits >, 1260
std::basic_istream< _CharT, _Traits >, 1307
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1348
std::basic_ofstream< _CharT, _Traits >, 1387
std::basic_ostream< _CharT, _Traits >, 1420
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1453
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1723
eofbit
std::basic_fstream< _CharT, _Traits >, 1180
std::basic_ifstream< _CharT, _Traits >, 1223
std::basic_ios< _CharT, _Traits >, 1248
std::basic_iostream< _CharT, _Traits >, 1295
std::basic_istream< _CharT, _Traits >, 1334
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1375
std::basic_ofstream< _CharT, _Traits >, 1408
std::basic_ostream< _CharT, _Traits >, 1440
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1473
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1757
std::ios_base, 2298
epptr
__gnu_cxx::enc_filebuf< _CharT >, 2116
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2941
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2961
std::basic_filebuf< _CharT, _Traits >, 1117
std::basic_streambuf< _CharT, _Traits >, 1491
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1698
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3254
epsilon
std::numeric_limits< _Tp >, 2599
eq_by_less.hpp, 3563
equal
Non-Mutating, 47–49
std::istreambuf_iterator< _CharT, _Traits >, 2327
equal_range
Binary Search, 86, 87
std::map< _Key, _Tp, _Compare, _Alloc >, 2422

std::multimap< _Key, _Tp, _Compare, _Alloc >, 2516
std::multiset< _Key, _Compare, _Alloc >, 2539
std::set< _Key, _Compare, _Alloc >, 2891
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3127, 3128
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3155
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3180
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3204
equally_split.h, 3655
equals
std::tr2::bool_set, 1814
equivalent
std::error_category, 2140
erase
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 847
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1526–1528
std::basic_string< _CharT, _Traits, _Alloc >, 1624–1626
std::deque< _Tp, _Alloc >, 2070
std::list< _Tp, _Alloc >, 2367, 2368
std::map< _Key, _Tp, _Compare, _Alloc >, 2423, 2424
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2516–2518
std::multiset< _Key, _Compare, _Alloc >, 2540, 2541
std::set< _Key, _Compare, _Alloc >, 2891, 2892
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3128–3130
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3155–3157
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3181, 3182
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3205, 3206
std::vector< _Tp, _Alloc >, 3237
erase_after
std::chrono::tzdb_list, 3086
std::forward_list< _Tp, _Alloc >, 2179
erase_can_throw
__gnu_pbds::container_traits< Cntnr >, 1933
erase_fn_imps.hpp, 3547, 3548
erase_if.h, 3321
erase_no_store_hash_fn_imps.hpp, 3560
erase_store_hash_fn_imps.hpp, 3560
error_backref
std::regex_constants, 715
error_badbrace
std::regex_constants, 715
error_badrepeat
std::regex_constants, 716
error_brace
std::regex_constants, 716
error_brack
std::regex_constants, 716
error_code
std::error_code, 2142
error_collate
std::regex_constants, 716
error_complexity
std::regex_constants, 716
error_condition
std::error_condition, 2143
error_constants.h, 3723
error_ctype
std::regex_constants, 716
error_escape
std::regex_constants, 716
error_paren
std::regex_constants, 716
error_range
std::regex_constants, 716
error_space
std::regex_constants, 717
error_stack
std::regex_constants, 717
error_type
std::regex_constants, 715
event
std::basic_fstream< _CharT, _Traits >, 1142
std::basic_ifstream< _CharT, _Traits >, 1193
std::basic_ios< _CharT, _Traits >, 1234
std::basic_iostream< _CharT, _Traits >, 1259
std::basic_istream< _CharT, _Traits >, 1305
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1345
std::basic_ofstream< _CharT, _Traits >, 1385
std::basic_ostream< _CharT, _Traits >, 1418
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1451
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1720
std::ios_base, 2291
event_callback
std::basic_fstream< _CharT, _Traits >, 1140
std::basic_ifstream< _CharT, _Traits >, 1191
std::basic_ios< _CharT, _Traits >, 1232
std::basic_iostream< _CharT, _Traits >, 1257
std::basic_istream< _CharT, _Traits >, 1303
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1343
std::basic_ofstream< _CharT, _Traits >, 1383
std::basic_ostream< _CharT, _Traits >, 1416

std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1449
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1719
std::ios_base, 2290
exception, 3728
exception.h, 3321
exception.hpp, 3581
exception_defines.h, 3321
exception_ptr.h, 3322
rethrow_exception, 3322
Exceptions, 133, 148
__verbose_terminate_handler, 135
current_exception, 135
get_terminate, 135
make_exception_ptr, 135
rethrow_exception, 135
rethrow_if_nested, 136
set_terminate, 136
terminate, 136
terminate_handler, 134
throw_with_nested, 136
uncaught_exception, 137
exceptions
std::basic_fstream< _CharT, _Traits >, 1145
std::basic_ifstream< _CharT, _Traits >, 1196
std::basic_ios< _CharT, _Traits >, 1236
std::basic_iostream< _CharT, _Traits >, 1260, 1261
std::basic_istream< _CharT, _Traits >, 1307
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1348
std::basic_ofstream< _CharT, _Traits >, 1388
std::basic_ostream< _CharT, _Traits >, 1420
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1453
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1723
exclusive_scan
Generalized Numeric operations, 8, 9
exp
Complex Numbers, 185
math.h, 3625
expected, 3530
expint
Mathematical Special Functions, 211
TR1 Mathematical Special Functions, 254
expintf
Mathematical Special Functions, 212
expintl
Mathematical Special Functions, 212
explicit
__gnu_cxx::pair< _T1, _T2 >, 2714
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 917–919
std::complex< _Tp >, 1905
std::pair< _T1, _T2 >, 2719, 2720
exponential_distribution
std::exponential_distribution< _RealType >, 2150
extc++.h, 3724
extended
std::regex_constants, 720
Extensions, 138
external_load_access
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1819
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2255
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2265
extptr_allocator.h, 3537

fabs
Complex Numbers, 185
math.h, 3626
std, 598
facet
std::locale::facet, 2157
fail
std::basic_fstream< _CharT, _Traits >, 1145
std::basic_ifstream< _CharT, _Traits >, 1196
std::basic_ios< _CharT, _Traits >, 1237
std::basic_iostream< _CharT, _Traits >, 1261
std::basic_istream< _CharT, _Traits >, 1308
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1349
std::basic_ofstream< _CharT, _Traits >, 1388
std::basic_ostream< _CharT, _Traits >, 1421
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1454
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1724
failbit
std::basic_fstream< _CharT, _Traits >, 1180
std::basic_ifstream< _CharT, _Traits >, 1223
std::basic_ios< _CharT, _Traits >, 1248
std::basic_iostream< _CharT, _Traits >, 1295
std::basic_istream< _CharT, _Traits >, 1334
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1375
std::basic_ofstream< _CharT, _Traits >, 1408
std::basic_ostream< _CharT, _Traits >, 1440
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1473
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1757
std::ios_base, 2298
failed

std::ostreambuf_iterator< _CharT, _Traits >, 2695
false_type
Metaprogramming, 359
falsename
std::numpunct< _CharT >, 2679
std::numpunct_byname< _CharT >, 2683
fd
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2942
features.h, 3655
_GLIBCXX_BAL_QUICKSORT, 3655
_GLIBCXX_FIND_CONSTANT_SIZE_BLOCKS, 3655
_GLIBCXX_FIND_EQUAL_SPLIT, 3656
_GLIBCXX_FIND_GROWING_BLOCKS, 3656
_GLIBCXX_MERGESORT, 3656
_GLIBCXX_QUICKSORT, 3656
_GLIBCXX_TREE_DYNAMIC_BALANCING, 3656
_GLIBCXX_TREE_FULL_COPY, 3656
_GLIBCXX_TREE_INITIAL_SPLITTING, 3657
fenv.h, 3607
file
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2942
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2962
file_time_type
Filesystem, 166
Filesystem TS, 284
file_type
Filesystem, 166
Filesystem TS, 285
filebuf
I/O, 169
Filesystem, 161
begin, 166
copy_options, 166
directory_options, 166
end, 166, 167
file_time_type, 166
file_type, 166
perm_options, 166
perms, 166
u8path, 167
filesystem, 3607
Filesystem TS, 280
copy_options, 285
directory_options, 285
file_time_type, 284
file_type, 285
perms, 285
fill
Mutating, 25
std::basic_fstream< _CharT, _Traits >, 1146
std::basic_ifstream< _CharT, _Traits >, 1197
std::basic_ios< _CharT, _Traits >, 1237
std::basic_iostream< _CharT, _Traits >, 1261, 1262
std::basic_istream< _CharT, _Traits >, 1308
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1349
std::basic_ofstream< _CharT, _Traits >, 1389
std::basic_ostream< _CharT, _Traits >, 1421
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1454
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1724
fill_minimal_n
__gnu_parallel::_Settings, 1032
fill_n
Mutating, 26
find
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 848, 849
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1528–1533
Non-Mutating, 49
std::basic_string< _CharT, _Traits, _Alloc >, 1626–1629
std::map< _Key, _Tp, _Compare, _Alloc >, 2425
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2518
std::multiset< _Key, _Compare, _Alloc >, 2541
std::set< _Key, _Compare, _Alloc >, 2893
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3130, 3131
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3157, 3158
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3182, 3183
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3206, 3207
find.h, 3657
find_by_order
__gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >, 3047
__gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3061
find_end
Non-Mutating, 50, 51
find_first
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2102
find_first_not_of
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 850, 851
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1533–1537
std::basic_string< _CharT, _Traits, _Alloc >, 1630–1633
find_first_of
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc,

_Base >, 852, 853
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1538–1542
Non-Mutating, 51, 52
std::basic_string< _CharT, _Traits, _Alloc >, 1633–1636
find_fn_imps.hpp, 3548, 3549
find_if
Non-Mutating, 53
find_if_not
Non-Mutating, 53
find_increasing_factor
__gnu_parallel::_Settings, 1032
find_initial_block_size
__gnu_parallel::_Settings, 1032
find_last_not_of
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 853–855
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1543–1547
std::basic_string< _CharT, _Traits, _Alloc >, 1636–1639
find_last_of
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 855–857
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1548–1552
std::basic_string< _CharT, _Traits, _Alloc >, 1640–1643
find_maximum_block_size
__gnu_parallel::_Settings, 1032
find_next
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2102
find_no_store_hash_fn_imps.hpp, 3563
find_scale_factor
__gnu_parallel::_Settings, 1032
find_selectors.h, 3657
find_sequential_search_size
__gnu_parallel::_Settings, 1032
find_store_hash_fn_imps.hpp, 3560
first
__gnu_cxx::pair< _T1, _T2 >, 2716
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 920
std::pair< _T1, _T2 >, 2721
first_argument_type
__gnu_cxx::equal_to< _Tp >, 2136
__gnu_cxx::project1st< _Arg1, _Arg2 >, 2781
__gnu_cxx::project2nd< _Arg1, _Arg2 >, 2781
__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >, 897
__gnu_parallel::_EqualTo< _T1, _T2 >, 898
__gnu_parallel::_Less< _T1, _T2 >, 924
__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >, 925
__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >, 927
__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >, 953
__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >, 965
std::binary_function< _Arg1, _Arg2, _Result >, 1778
std::binary_negate< _Predicate >, 1788
std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >, 1915
std::const_mem_fun1_t< _Ret, _Tp, _Arg >, 1916
std::divides< _Tp >, 2092
std::divides< void >, 2093
std::equal_to< _Tp >, 2137
std::equal_to< void >, 2139
std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >, 2707
std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >, 2709
std::greater< _Tp >, 2227
std::greater< void >, 2229
std::greater_equal< _Tp >, 2230
std::greater_equal< void >, 2231
std::less< _Tp >, 2344
std::less< void >, 2346
std::less_equal< _Tp >, 2347
std::less_equal< void >, 2348
std::logical_and< _Tp >, 2392
std::logical_and< void >, 2393
std::logical_or< _Tp >, 2396
std::logical_or< void >, 2397
std::mem_fun1_ref_t< _Ret, _Tp, _Arg >, 2449
std::mem_fun1_t< _Ret, _Tp, _Arg >, 2450
std::minus< _Tp >, 2465
std::minus< void >, 2466
std::modulus< _Tp >, 2468
std::modulus< void >, 2469
std::multiplies< _Tp >, 2527
std::multiplies< void >, 2528
std::not_equal_to< _Tp >, 2564
std::not_equal_to< void >, 2566
std::owner_less< shared_ptr< _Tp > >, 2708
std::owner_less< void >, 2709
std::owner_less< weak_ptr< _Tp > >, 2710
std::plus< _Tp >, 2760
std::pointer_to_binary_function< _Arg1, _Arg2, _Result >, 2763
first_type
__gnu_cxx::pair< _T1, _T2 >, 2713
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 917
std::pair< _T1, _T2 >, 2719
fixed

std, 598
std::basic_fstream< _CharT, _Traits >, 1181
std::basic_ifstream< _CharT, _Traits >, 1224
std::basic_ios< _CharT, _Traits >, 1248
std::basic_iostream< _CharT, _Traits >, 1295
std::basic_istream< _CharT, _Traits >, 1335
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1375
std::basic_ofstream< _CharT, _Traits >, 1409
std::basic_ostream< _CharT, _Traits >, 1440
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1473
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1758
std::ios_base, 2299
flags
std::basic_fstream< _CharT, _Traits >, 1146
std::basic_ifstream< _CharT, _Traits >, 1197
std::basic_ios< _CharT, _Traits >, 1238
std::basic_iostream< _CharT, _Traits >, 1262
std::basic_istream< _CharT, _Traits >, 1309
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1349, 1350
std::basic_ofstream< _CharT, _Traits >, 1389
std::basic_ostream< _CharT, _Traits >, 1422
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1455
std::basic_regex< _Ch_type, _Rx_traits >, 1484
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1724, 1725
std::ios_base, 2292
flat_map, 3607
flat_set, 3607
flip
std::bitset< _Nb >, 1807
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2102
float_denorm_style
std, 589
float_round_style
std, 589
floatfield
std::basic_fstream< _CharT, _Traits >, 1181
std::basic_ifstream< _CharT, _Traits >, 1224
std::basic_ios< _CharT, _Traits >, 1248
std::basic_iostream< _CharT, _Traits >, 1295
std::basic_istream< _CharT, _Traits >, 1335
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1375
std::basic_ofstream< _CharT, _Traits >, 1409
std::basic_ostream< _CharT, _Traits >, 1440
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1473
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1758
std::ios_base, 2299
floor
Time, 377, 378
flush
std, 598
std::basic_fstream< _CharT, _Traits >, 1147
std::basic_iostream< _CharT, _Traits >, 1263
std::basic_ofstream< _CharT, _Traits >, 1390
std::basic_ostream< _CharT, _Traits >, 1422
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1455
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1725
fmtflags
std::basic_fstream< _CharT, _Traits >, 1140
std::basic_ifstream< _CharT, _Traits >, 1191
std::basic_ios< _CharT, _Traits >, 1232
std::basic_iostream< _CharT, _Traits >, 1257
std::basic_istream< _CharT, _Traits >, 1304
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1344
std::basic_ofstream< _CharT, _Traits >, 1383
std::basic_ostream< _CharT, _Traits >, 1417
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1449
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1719
std::ios_base, 2290
for_each
Non-Mutating, 54
for_each.h, 3658
for_each_minimal_n
__gnu_parallel::_Settings, 1032
for_each_n
Non-Mutating, 54
for_each_selectors.h, 3658
format, 3607
std::filesystem::path, 2747
std::match_results< _Bi_iter, _Alloc >, 2442
format_default
std::regex_constants, 720
format_first_only
std::regex_constants, 721
format_no_copy
std::regex_constants, 721
format_sed
std::regex_constants, 721
formatfwd.h, 3322
formatter.h, 3504
forward
Utilities, 306
forward_as_tuple
Utilities, 307
forward_like

Utilities, 308
forward_list, 3607–3609
std::forward_list< _Tp, _Alloc >, 2172–2175
forward_list.h, 3322
forward_list.tcc, 3323
fpos
std::fpos< _StateT >, 2189
frac_digits
std::moneypunct< _CharT, _Intl >, 2486
std::moneypunct_byname< _CharT, _Intl >, 2494
from_bytes
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3273
from_chars
std, 599
front
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 857
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1552
std::basic_string< _CharT, _Traits, _Alloc >, 1643
std::chrono::tzdb_list, 3086
std::deque< _Tp, _Alloc >, 2070, 2071
std::forward_list< _Tp, _Alloc >, 2180
std::list< _Tp, _Alloc >, 2368
std::queue< _Tp, _Sequence >, 2787
std::vector< _Tp, _Alloc >, 3237, 3238
front_insert_iterator
std::front_insert_iterator< _Container >, 2193
front_inserter
Iterators, 175
fs_dir.h, 3324
fs_fwd.h, 3325, 3326
fs_ops.h, 3328, 3330
fs_path.h, 3333
fstream, 3609
I/O, 169
fstream.tcc, 3334
funcref_impl.h, 3334
functexcept.h, 3334
function
std::function< _Res(_ArgTypes...)>, 2195, 2196
Function Objects, 315
mem_fn, 316
function_ref
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2201, 2202
functional, 3610, 3611
functional_hash.h, 3334
functions.h, 3505
funcwrap.h, 3335
future, 3613
std::future< _Res >, 2204
std::future< _Res & >, 2205
std::future< void >, 2207
future_category
Futures, 120
future_errc
Futures, 119
future_status
Futures, 119
Futures, 118
async, 120
future_category, 120
future_errc, 119
future_status, 119
launch, 119
make_error_code, 120
make_error_condition, 120
swap, 121

gamma_distribution
std::gamma_distribution< _RealType >, 2210
gbump
__gnu_cxx::enc_filebuf< _CharT >, 2116
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2942
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2962
std::basic_filebuf< _CharT, _Traits >, 1117
std::basic_streambuf< _CharT, _Traits >, 1491
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1698
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3254
gcd
std::experimental, 706
gcount
std::basic_fstream< _CharT, _Traits >, 1147
std::basic_ifstream< _CharT, _Traits >, 1198
std::basic_iostream< _CharT, _Traits >, 1263
std::basic_istream< _CharT, _Traits >, 1309
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1350
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1725
Generalized Numeric operations, 5
accumulate, 6, 7
adjacent_difference, 7, 8
exclusive_scan, 8, 9
inclusive_scan, 10, 11
inner_product, 12
iota, 13
partial_sum, 14
reduce, 15, 16
transform_exclusive_scan, 16
transform_inclusive_scan, 17, 18
transform_reduce, 19, 20
generate
Mutating, 26
generate_canonical

Random Number Generation, 237
std, 599
generate_minimal_n
__gnu_parallel::_Settings, 1032
generate_n
Mutating, 27
generator, 3614
generic_category
Diagnostics, 130
get
__gnu_parallel::_Settings, 1031
std, 599–601
std::auto_ptr< _Tp >, 1095
std::basic_fstream< _CharT, _Traits >, 1147–1149
std::basic_ifstream< _CharT, _Traits >, 1198–1200
std::basic_iostream< _CharT, _Traits >, 1263–1265
std::basic_istream< _CharT, _Traits >, 1310–1312
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1350–1352
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1725–1727
std::future< _Res >, 2204
std::future< _Res & >, 2205
std::future< void >, 2207
std::money_get< _CharT, _InIter >, 2473, 2474
std::num_get< _CharT, _InIter >, 2577–2584
std::shared_future< _Res >, 2901
std::shared_future< _Res & >, 2902
std::shared_ptr< _Tp >, 2914
std::time_get< _CharT, _InIter >, 3016, 3017
std::time_get_byname< _CharT, _InIter >, 3027, 3028
std::unique_ptr< _Tp, _Dp >, 3102
std::unique_ptr< _Tp[], _Dp >, 3107
Utilities, 308
get_actual_size
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2266
get_allocator
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 857
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1553
std::basic_string< _CharT, _Traits, _Alloc >, 1644
std::deque< _Tp, _Alloc >, 2071
std::forward_list< _Tp, _Alloc >, 2180
std::list< _Tp, _Alloc >, 2368
std::map< _Key, _Tp, _Compare, _Alloc >, 2426
std::match_results< _Bi_iter, _Alloc >, 2442
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2519
std::multiset< _Key, _Compare, _Alloc >, 2542
std::set< _Key, _Compare, _Alloc >, 2893
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2103
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3131
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3158
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3183
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3207
std::vector< _Tp, _Alloc >, 3238
std::vector< bool, _Alloc >, 3249
get_child
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 957
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 960
get_comb_hash_fn
__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1831
get_comb_probe_fn
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2224
get_date
std::time_get< _CharT, _InIter >, 3017
std::time_get_byname< _CharT, _InIter >, 3029
get_default_resource
memory_resource, 3634
get_deleter
Pointer Abstractions, 348
std::experimental, 706
std::unique_ptr< _Tp, _Dp >, 3102
std::unique_ptr< _Tp[], _Dp >, 3107
get_eq_fn
__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1832
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2225
get_hash_fn
__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1832
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2225
get_id

std::this_thread, 725
get_l_child
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1767
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1772
__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >, 2702
__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >, 2704
get_load
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1820
get_loads
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2255
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2266
get_metadata
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1768
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1772
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 957
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 960
get_money
std, 601
get_monthname
std::time_get< _CharT, _InIter >, 3018
std::time_get_byname< _CharT, _InIter >, 3029
get_nearest_larger_size
__gnu_pbds::sample_size_policy, 2865
get_nearest_smaller_size
__gnu_pbds::sample_size_policy, 2865
get_new_handler
std, 601
get_new_size
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2266
__gnu_pbds::sample_resize_policy, 2860
get_probe_fn
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2225
get_r_child
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1768
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1772
__gnu_pbds::detail::ov_tree_node_const_it_< Value_Type, Metadata_Type, _Alloc >, 2702
__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >, 2704
get_resize_policy
__gnu_pbds::detail::cc_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Hash_Fn, Resize_Policy >, 1832
__gnu_pbds::detail::gp_ht_map< Key, Mapped, Hash_Fn, Eq_Fn, _Alloc, Store_Hash, Comb_Probe_Fn, Probe_Fn, Resize_Policy >, 2225, 2226
get_size_policy
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2267
get_temporary_buffer
std, 601
get_terminate
Exceptions, 135
get_time
std, 602
std::time_get< _CharT, _InIter >, 3019
std::time_get_byname< _CharT, _InIter >, 3030
get_trigger_policy
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2267
get_weekday
std::time_get< _CharT, _InIter >, 3019
std::time_get_byname< _CharT, _InIter >, 3030
get_year
std::time_get< _CharT, _InIter >, 3020
std::time_get_byname< _CharT, _InIter >, 3031
getline
std, 602–604
std::basic_fstream< _CharT, _Traits >, 1150
std::basic_ifstream< _CharT, _Traits >, 1201
std::basic_iostream< _CharT, _Traits >, 1266
std::basic_istream< _CharT, _Traits >, 1312, 1313
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1353
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1728
getloc
__gnu_cxx::enc_filebuf< _CharT >, 2116
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2942
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2962
std::basic_filebuf< _CharT, _Traits >, 1117
std::basic_fstream< _CharT, _Traits >, 1151
std::basic_ifstream< _CharT, _Traits >, 1202

std::basic_ios< _CharT, _Traits >, 1238
std::basic_iostream< _CharT, _Traits >, 1267
std::basic_istream< _CharT, _Traits >, 1313
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1354
std::basic_ofstream< _CharT, _Traits >, 1390
std::basic_ostream< _CharT, _Traits >, 1423
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1456
std::basic_regex< _Ch_type, _Rx_traits >, 1484
std::basic_streambuf< _CharT, _Traits >, 1492
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1698
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1729
std::ios_base, 2292
std::regex_traits< _Ch_type >, 2835
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3255
global
std::locale, 2385
good
std::basic_fstream< _CharT, _Traits >, 1151
std::basic_ifstream< _CharT, _Traits >, 1202
std::basic_ios< _CharT, _Traits >, 1238
std::basic_iostream< _CharT, _Traits >, 1267
std::basic_istream< _CharT, _Traits >, 1314
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1354
std::basic_ofstream< _CharT, _Traits >, 1390
std::basic_ostream< _CharT, _Traits >, 1423
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1456
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1729
goodbit
std::basic_fstream< _CharT, _Traits >, 1181
std::basic_ifstream< _CharT, _Traits >, 1224
std::basic_ios< _CharT, _Traits >, 1248
std::basic_iostream< _CharT, _Traits >, 1295
std::basic_istream< _CharT, _Traits >, 1335
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1375
std::basic_ofstream< _CharT, _Traits >, 1409
std::basic_ostream< _CharT, _Traits >, 1440
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1474
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1758
std::ios_base, 2299
gp_hash_table
__gnu_pbds::gp_hash_table< Key, Mapped, Hash_Fn, Eq_Fn, Comb_Probe_Fn, Probe_Fn, Resize_Policy, Store_Hash, _Alloc >, 2216–2220
gp_ht_map_.hpp, 3563
gptr
__gnu_cxx::enc_filebuf< _CharT >, 2116
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2943
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2962
std::basic_filebuf< _CharT, _Traits >, 1117
std::basic_streambuf< _CharT, _Traits >, 1492
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1699
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3255
grep
std::regex_constants, 721
grouping
std::moneypunct< _CharT, _Intl >, 2487
std::moneypunct_byname< _CharT, _Intl >, 2494
std::numpunct< _CharT >, 2679
std::numpunct_byname< _CharT >, 2684
gslice
Numeric Arrays, 222
gslice.h, 3336
gslice_array
Numeric Arrays, 223
gslice_array.h, 3336

has_denorm
std::__numeric_limits_base, 807
std::numeric_limits< _Tp >, 2601
std::numeric_limits< bool >, 2605
std::numeric_limits< char >, 2609
std::numeric_limits< char16_t >, 2613
std::numeric_limits< char32_t >, 2617
std::numeric_limits< double >, 2621
std::numeric_limits< float >, 2625
std::numeric_limits< int >, 2629
std::numeric_limits< long >, 2633
std::numeric_limits< long double >, 2637
std::numeric_limits< long long >, 2641
std::numeric_limits< short >, 2645
std::numeric_limits< signed char >, 2649
std::numeric_limits< unsigned char >, 2653
std::numeric_limits< unsigned int >, 2657
std::numeric_limits< unsigned long >, 2661
std::numeric_limits< unsigned long long >, 2665
std::numeric_limits< unsigned short >, 2669
std::numeric_limits< wchar_t >, 2673
has_denorm_loss
std::__numeric_limits_base, 807
std::numeric_limits< _Tp >, 2601
std::numeric_limits< bool >, 2605
std::numeric_limits< char >, 2609
std::numeric_limits< char16_t >, 2613
std::numeric_limits< char32_t >, 2617
std::numeric_limits< double >, 2621
std::numeric_limits< float >, 2625
std::numeric_limits< int >, 2629

std::numeric_limits< long >, 2633
std::numeric_limits< long double >, 2637
std::numeric_limits< long long >, 2641
std::numeric_limits< short >, 2645
std::numeric_limits< signed char >, 2649
std::numeric_limits< unsigned char >, 2653
std::numeric_limits< unsigned int >, 2657
std::numeric_limits< unsigned long >, 2661
std::numeric_limits< unsigned long long >, 2665
std::numeric_limits< unsigned short >, 2669
std::numeric_limits< wchar_t >, 2673
has_facet
Locales, 178
std::locale, 2387
std::locale::id, 2269
has_infinity
std::__numeric_limits_base, 808
std::numeric_limits< _Tp >, 2601
std::numeric_limits< bool >, 2605
std::numeric_limits< char >, 2609
std::numeric_limits< char16_t >, 2613
std::numeric_limits< char32_t >, 2617
std::numeric_limits< double >, 2621
std::numeric_limits< float >, 2625
std::numeric_limits< int >, 2629
std::numeric_limits< long >, 2633
std::numeric_limits< long double >, 2637
std::numeric_limits< long long >, 2641
std::numeric_limits< short >, 2645
std::numeric_limits< signed char >, 2649
std::numeric_limits< unsigned char >, 2653
std::numeric_limits< unsigned int >, 2657
std::numeric_limits< unsigned long >, 2661
std::numeric_limits< unsigned long long >, 2665
std::numeric_limits< unsigned short >, 2669
std::numeric_limits< wchar_t >, 2673
has_quiet_NaN
std::__numeric_limits_base, 808
std::numeric_limits< _Tp >, 2601
std::numeric_limits< bool >, 2605
std::numeric_limits< char >, 2609
std::numeric_limits< char16_t >, 2613
std::numeric_limits< char32_t >, 2617
std::numeric_limits< double >, 2621
std::numeric_limits< float >, 2625
std::numeric_limits< int >, 2629
std::numeric_limits< long >, 2633
std::numeric_limits< long double >, 2637
std::numeric_limits< long long >, 2641
std::numeric_limits< short >, 2645
std::numeric_limits< signed char >, 2649
std::numeric_limits< unsigned char >, 2653
std::numeric_limits< unsigned int >, 2657
std::numeric_limits< unsigned long >, 2661
std::numeric_limits< unsigned long long >, 2665
std::numeric_limits< unsigned short >, 2669
std::numeric_limits< wchar_t >, 2673
has_signaling_NaN
std::__numeric_limits_base, 808
std::numeric_limits< _Tp >, 2601
std::numeric_limits< bool >, 2605
std::numeric_limits< char >, 2609
std::numeric_limits< char16_t >, 2613
std::numeric_limits< char32_t >, 2617
std::numeric_limits< double >, 2621
std::numeric_limits< float >, 2625
std::numeric_limits< int >, 2629
std::numeric_limits< long >, 2633
std::numeric_limits< long double >, 2637
std::numeric_limits< long long >, 2641
std::numeric_limits< short >, 2645
std::numeric_limits< signed char >, 2649
std::numeric_limits< unsigned char >, 2653
std::numeric_limits< unsigned int >, 2657
std::numeric_limits< unsigned long >, 2661
std::numeric_limits< unsigned long long >, 2665
std::numeric_limits< unsigned short >, 2669
std::numeric_limits< wchar_t >, 2673
hash
std::__cxx11::collate< _CharT >, 1898
Hash-Based, 146
hash_bytes.h, 3336
hash_eq_fn.hpp, 3563
hash_exponential_size_policy
__gnu_pbds::hash_exponential_size_policy< Size_Type >, 2253
hash_exponential_size_policy_imp.hpp, 3575
hash_fun.h, 3283
hash_function
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3131
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3158
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3183
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3207
hash_load_check_resize_trigger
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2255
hash_load_check_resize_trigger_imp.hpp, 3575
hash_load_check_resize_trigger_size_base.hpp, 3575
hash_map, 3284
hash_policy.hpp, 3582
hash_prime_size_policy
__gnu_pbds::hash_prime_size_policy, 2262
hash_prime_size_policy_imp.hpp, 3575
hash_set, 3284

hash_standard_resize_policy
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2266
hash_standard_resize_policy_imp.hpp, 3575
hasher
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3117
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3146
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3171
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3195
Hashes, 323
hashtable.h, 3337
hashtable_policy.h, 3337
Heap, 90
is_heap, 91
is_heap_until, 91, 92
make_heap, 92, 93
pop_heap, 93, 94
push_heap, 94
sort_heap, 95
Heap-Based, 147
helper_functions.h, 3507
hermite
Mathematical Special Functions, 212
TR1 Mathematical Special Functions, 254
hermitef
Mathematical Special Functions, 212
hermitel
Mathematical Special Functions, 213
hex
std, 604
std::basic_fstream< _CharT, _Traits >, 1181
std::basic_ifstream< _CharT, _Traits >, 1224
std::basic_ios< _CharT, _Traits >, 1249
std::basic_iostream< _CharT, _Traits >, 1296
std::basic_istream< _CharT, _Traits >, 1335
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1376
std::basic_ofstream< _CharT, _Traits >, 1409
std::basic_ostream< _CharT, _Traits >, 1441
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1474
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1758
std::ios_base, 2299
hexfloat
std, 604
high_resolution_clock
Time, 374
hours
Time, 374
hyperg
Mathematical Special Functions, 213
std::tr1, 730
hypergf
Mathematical Special Functions, 213
std::tr1, 730
hypergl
Mathematical Special Functions, 214
std::tr1, 731

I/O, 167
filebuf, 169
fstream, 169
ifstream, 169
ios, 169
iostream, 169
istream, 169
istringstream, 169
ofstream, 169
ostream, 169
ostringstream, 170
streambuf, 170
stringbuf, 170
stringstream, 170
wfilebuf, 170
wfstream, 170
wifstream, 170
wios, 170
wiostream, 170
wistream, 170
wistringstream, 170
wofstream, 171
wostream, 171
wostringstream, 171
wstreambuf, 171
wstringbuf, 171
wstringstream, 171
icase
std::regex_constants, 721
id
std::__cxx11::collate< _CharT >, 1899
std::ctype< _CharT >, 1957
std::ctype< char >, 1987
std::ctype< wchar_t >, 2001
std::ctype_byname< _CharT >, 2016
std::ctype_byname< char >, 2039
std::locale::id, 2269
std::messages< _CharT >, 2461
std::messages_byname< _CharT >, 2464
std::money_get< _CharT, _InIter >, 2475
std::money_put< _CharT, _OutIter >, 2480
std::moneypunct< _CharT, _Intl >, 2489
std::moneypunct_byname< _CharT, _Intl >, 2496

std::num_get< _CharT, _InIter >, 2585
std::num_put< _CharT, _OutIter >, 2598
std::numpunct< _CharT >, 2680
std::numpunct_byname< _CharT >, 2685
std::time_get< _CharT, _InIter >, 3021
std::time_get_byname< _CharT, _InIter >, 3032
std::time_put< _CharT, _OutIter >, 3037
std::time_put_byname< _CharT, _OutIter >, 3040
identity_element
SGI, 159
ifstream
I/O, 169
ignore
std, 649
std::basic_fstream< _CharT, _Traits >, 1152
std::basic_ifstream< _CharT, _Traits >, 1202, 1203
std::basic_iostream< _CharT, _Traits >, 1267, 1268
std::basic_istream< _CharT, _Traits >, 1314, 1315
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1354, 1355
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1730
imbue
__gnu_cxx::enc_filebuf< _CharT >, 2117
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2943
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2963
std::basic_filebuf< _CharT, _Traits >, 1118
std::basic_fstream< _CharT, _Traits >, 1153
std::basic_ifstream< _CharT, _Traits >, 1203
std::basic_ios< _CharT, _Traits >, 1239
std::basic_iostream< _CharT, _Traits >, 1268
std::basic_istream< _CharT, _Traits >, 1315
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1356
std::basic_ofstream< _CharT, _Traits >, 1390
std::basic_ostream< _CharT, _Traits >, 1423
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1456
std::basic_regex< _Ch_type, _Rx_traits >, 1484
std::basic_streambuf< _CharT, _Traits >, 1492
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1699
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1731
std::ios_base, 2293
std::regex_traits< _Ch_type >, 2835
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3255
in
std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 761
std::basic_fstream< _CharT, _Traits >, 1181
std::basic_ifstream< _CharT, _Traits >, 1224
std::basic_ios< _CharT, _Traits >, 1249
std::basic_iostream< _CharT, _Traits >, 1296
std::basic_istream< _CharT, _Traits >, 1335
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1376
std::basic_ofstream< _CharT, _Traits >, 1409
std::basic_ostream< _CharT, _Traits >, 1441
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1474
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1758
std::codecvt< _InternT, _ExternT, _StateT >, 1848
std::codecvt< _InternT, _ExternT, encoding_state >, 1853, 1854
std::codecvt< char, char, mbstate_t >, 1861
std::codecvt< char16_t, char, mbstate_t >, 1868, 1869
std::codecvt< char32_t, char, mbstate_t >, 1875, 1876
std::codecvt< wchar_t, char, mbstate_t >, 1883, 1884
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1892
std::ios_base, 2299
in_avail
__gnu_cxx::enc_filebuf< _CharT >, 2117
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2943
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2963
std::basic_filebuf< _CharT, _Traits >, 1118
std::basic_streambuf< _CharT, _Traits >, 1493
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1699
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3256
in_place
Optional values, 293
includes
Set Operations, 97
inclusive_scan
Generalized Numeric operations, 10, 11
increment
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2354
independent_bits_engine
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2272, 2273
index_sequence
std, 587
index_sequence_for
std, 587
indirect.h, 3338
indirect_array
Numeric Arrays, 223
indirect_array.h, 3338
infinity
std::numeric_limits< _Tp >, 2599
info_fn_imps.hpp, 3549, 3550

init
std::basic_fstream< _CharT, _Traits >, 1153
std::basic_ifstream< _CharT, _Traits >, 1204
std::basic_ios< _CharT, _Traits >, 1239
std::basic_iostream< _CharT, _Traits >, 1269
std::basic_istream< _CharT, _Traits >, 1316
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1356
std::basic_ofstream< _CharT, _Traits >, 1391
std::basic_ostream< _CharT, _Traits >, 1423
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1456
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1731
initializer_list, 3729
inner_product
Generalized Numeric operations, 12
inout_ptr
std, 604
inplace_merge
Sorting, 66
inplace_tags.h, 3338
inplace_vector, 3614
insert
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 857–862
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1553–1559
std::basic_string< _CharT, _Traits, _Alloc >, 1644–1653
std::deque< _Tp, _Alloc >, 2071–2073
std::list< _Tp, _Alloc >, 2368–2370
std::map< _Key, _Tp, _Compare, _Alloc >, 2426–2429
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2519–2522
std::multiset< _Key, _Compare, _Alloc >, 2542, 2543
std::set< _Key, _Compare, _Alloc >, 2894, 2895
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3131–3134
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3158–3161
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3183–3185
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3207–3210
std::vector< _Tp, _Alloc >, 3238–3240
insert_after
std::forward_list< _Tp, _Alloc >, 2180, 2181
insert_fn_imps.hpp, 3550–3552
insert_iterator
std::insert_iterator< _Container >, 2283
insert_join_fn_imps.hpp, 3571
insert_no_store_hash_fn_imps.hpp, 3560, 3561
insert_store_hash_fn_imps.hpp, 3561
inserter
Iterators, 176
int_type
std::basic_ios< _CharT, _Traits >, 1233
std::basic_streambuf< _CharT, _Traits >, 1489
std::istreambuf_iterator< _CharT, _Traits >, 2326
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3252
intcmp.h, 3339
internal
std, 605
std::basic_fstream< _CharT, _Traits >, 1181
std::basic_ifstream< _CharT, _Traits >, 1225
std::basic_ios< _CharT, _Traits >, 1249
std::basic_iostream< _CharT, _Traits >, 1296
std::basic_istream< _CharT, _Traits >, 1335
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1376
std::basic_ofstream< _CharT, _Traits >, 1409
std::basic_ostream< _CharT, _Traits >, 1441
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1474
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1759
std::ios_base, 2300
intervals
std::piecewise_constant_distribution< _RealType >, 2754
std::piecewise_linear_distribution< _RealType >, 2758
intl
std::moneypunct< _CharT, _Intl >, 2489
Invalidation Guarantees, 150
invoke.h, 3339
io_errc
std, 589
iomanip, 3615
ios, 3616
I/O, 169
ios_base.h, 3340
iosfwd, 3617
iosfwd_file.h, 3341
iosfwd_span.h, 3342
iosfwd_string.h, 3342
iosfwd_sync.h, 3342
iostate
std::basic_fstream< _CharT, _Traits >, 1141
std::basic_ifstream< _CharT, _Traits >, 1192
std::basic_ios< _CharT, _Traits >, 1233
std::basic_iostream< _CharT, _Traits >, 1258
std::basic_istream< _CharT, _Traits >, 1304
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1344
std::basic_ofstream< _CharT, _Traits >, 1384

std::basic_ostream< _CharT, _Traits >, 1417
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1450
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1720
std::ios_base, 2291
iostream, 3617
I/O, 169
iota
__gnu_cxx, 403
Generalized Numeric operations, 13
is
std::__ctype_abstract_base< _CharT >, 774, 775
std::ctype< _CharT >, 1951, 1952
std::ctype< char >, 1972–1974
std::ctype< wchar_t >, 1996
std::ctype_byname< _CharT >, 2010, 2011
std::ctype_byname< char >, 2029, 2030
is_always_equal
__gnu_cxx::__alloc_traits< _Alloc, typename >, 750
std::allocator_traits< _Alloc >, 1056
std::allocator_traits< allocator< _Tp > >, 1061
std::allocator_traits< allocator< void > >, 1066
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1070
is_am
Time, 378
is_bind_expression_v
std::experimental, 707
is_bounded
std::__numeric_limits_base, 808
std::numeric_limits< _Tp >, 2601
std::numeric_limits< bool >, 2605
std::numeric_limits< char >, 2609
std::numeric_limits< char16_t >, 2613
std::numeric_limits< char32_t >, 2617
std::numeric_limits< double >, 2621
std::numeric_limits< float >, 2625
std::numeric_limits< int >, 2629
std::numeric_limits< long >, 2633
std::numeric_limits< long double >, 2637
std::numeric_limits< long long >, 2641
std::numeric_limits< short >, 2645
std::numeric_limits< signed char >, 2649
std::numeric_limits< unsigned char >, 2653
std::numeric_limits< unsigned int >, 2657
std::numeric_limits< unsigned long >, 2661
std::numeric_limits< unsigned long long >, 2665
std::numeric_limits< unsigned short >, 2669
std::numeric_limits< wchar_t >, 2673
is_corresponding_member
Metaprogramming, 361
is_detected
Detection idiom, 290
is_detected_convertible
Detection idiom, 291
is_detected_convertible_v
Detection idiom, 291
is_detected_exact
Detection idiom, 291
is_detected_exact_v
Detection idiom, 291
is_detected_v
Detection idiom, 291
is_emptyset
std::tr2::bool_set, 1814
is_exact
std::__numeric_limits_base, 808
std::numeric_limits< _Tp >, 2601
std::numeric_limits< bool >, 2605
std::numeric_limits< char >, 2609
std::numeric_limits< char16_t >, 2613
std::numeric_limits< char32_t >, 2617
std::numeric_limits< double >, 2621
std::numeric_limits< float >, 2625
std::numeric_limits< int >, 2629
std::numeric_limits< long >, 2633
std::numeric_limits< long double >, 2637
std::numeric_limits< long long >, 2641
std::numeric_limits< short >, 2645
std::numeric_limits< signed char >, 2649
std::numeric_limits< unsigned char >, 2653
std::numeric_limits< unsigned int >, 2657
std::numeric_limits< unsigned long >, 2661
std::numeric_limits< unsigned long long >, 2665
std::numeric_limits< unsigned short >, 2669
std::numeric_limits< wchar_t >, 2673
is_grow_needed
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1820
__gnu_pbds::sample_resize_trigger, 2863
is_heap
Heap, 91
is_heap_until
Heap, 91, 92
is_iec559
std::__numeric_limits_base, 808
std::numeric_limits< _Tp >, 2601
std::numeric_limits< bool >, 2605
std::numeric_limits< char >, 2609
std::numeric_limits< char16_t >, 2613
std::numeric_limits< char32_t >, 2617
std::numeric_limits< double >, 2621
std::numeric_limits< float >, 2625
std::numeric_limits< int >, 2629
std::numeric_limits< long >, 2633
std::numeric_limits< long double >, 2637
std::numeric_limits< long long >, 2641

std::numeric_limits< short >, 2645
std::numeric_limits< signed char >, 2649
std::numeric_limits< unsigned char >, 2653
std::numeric_limits< unsigned int >, 2657
std::numeric_limits< unsigned long >, 2661
std::numeric_limits< unsigned long long >, 2665
std::numeric_limits< unsigned short >, 2669
std::numeric_limits< wchar_t >, 2673
is_indeterminate
std::tr2::bool_set, 1814
is_integer
std::__numeric_limits_base, 808
std::numeric_limits< _Tp >, 2601
std::numeric_limits< bool >, 2605
std::numeric_limits< char >, 2609
std::numeric_limits< char16_t >, 2613
std::numeric_limits< char32_t >, 2617
std::numeric_limits< double >, 2621
std::numeric_limits< float >, 2625
std::numeric_limits< int >, 2629
std::numeric_limits< long >, 2633
std::numeric_limits< long double >, 2637
std::numeric_limits< long long >, 2641
std::numeric_limits< short >, 2645
std::numeric_limits< signed char >, 2649
std::numeric_limits< unsigned char >, 2653
std::numeric_limits< unsigned int >, 2657
std::numeric_limits< unsigned long >, 2661
std::numeric_limits< unsigned long long >, 2665
std::numeric_limits< unsigned short >, 2669
std::numeric_limits< wchar_t >, 2673
is_layout_compatible_v
Metaprogramming, 362
Variable Templates, 363
is_modulo
std::__numeric_limits_base, 808
std::numeric_limits< _Tp >, 2602
std::numeric_limits< bool >, 2606
std::numeric_limits< char >, 2610
std::numeric_limits< char16_t >, 2614
std::numeric_limits< char32_t >, 2618
std::numeric_limits< double >, 2622
std::numeric_limits< float >, 2626
std::numeric_limits< int >, 2630
std::numeric_limits< long >, 2634
std::numeric_limits< long double >, 2638
std::numeric_limits< long long >, 2642
std::numeric_limits< short >, 2646
std::numeric_limits< signed char >, 2650
std::numeric_limits< unsigned char >, 2654
std::numeric_limits< unsigned int >, 2658
std::numeric_limits< unsigned long >, 2662
std::numeric_limits< unsigned long long >, 2666
std::numeric_limits< unsigned short >, 2670
std::numeric_limits< wchar_t >, 2674
is_open
__gnu_cxx::enc_filebuf< _CharT >, 2117
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2944
std::basic_filebuf< _CharT, _Traits >, 1118
std::basic_fstream< _CharT, _Traits >, 1154
std::basic_ifstream< _CharT, _Traits >, 1204
std::basic_ofstream< _CharT, _Traits >, 1391
is_partitioned
Mutating, 27
is_permutation
Non-Mutating, 55, 56
is_placeholder_v
std::experimental, 707
is_pm
Time, 378
is_pointer_interconvertible_base_of_v
Metaprogramming, 362
Variable Templates, 363
is_pointer_interconvertible_with_class
Metaprogramming, 361
is_resize_needed
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1820
__gnu_pbds::sample_resize_policy, 2860
__gnu_pbds::sample_resize_trigger, 2863
is_signed
std::__numeric_limits_base, 808
std::numeric_limits< _Tp >, 2602
std::numeric_limits< bool >, 2606
std::numeric_limits< char >, 2610
std::numeric_limits< char16_t >, 2614
std::numeric_limits< char32_t >, 2618
std::numeric_limits< double >, 2622
std::numeric_limits< float >, 2626
std::numeric_limits< int >, 2630
std::numeric_limits< long >, 2634
std::numeric_limits< long double >, 2638
std::numeric_limits< long long >, 2642
std::numeric_limits< short >, 2646
std::numeric_limits< signed char >, 2650
std::numeric_limits< unsigned char >, 2654
std::numeric_limits< unsigned int >, 2658
std::numeric_limits< unsigned long >, 2662
std::numeric_limits< unsigned long long >, 2666
std::numeric_limits< unsigned short >, 2670
std::numeric_limits< wchar_t >, 2674
is_singleton
std::tr2::bool_set, 1814
is_sorted
Sorting, 67
is_sorted_until
Sorting, 68
is_specialized

std::__numeric_limits_base, 808
std::numeric_limits< _Tp >, 2602
std::numeric_limits< bool >, 2606
std::numeric_limits< char >, 2610
std::numeric_limits< char16_t >, 2614
std::numeric_limits< char32_t >, 2618
std::numeric_limits< double >, 2622
std::numeric_limits< float >, 2626
std::numeric_limits< int >, 2630
std::numeric_limits< long >, 2634
std::numeric_limits< long double >, 2638
std::numeric_limits< long long >, 2642
std::numeric_limits< short >, 2646
std::numeric_limits< signed char >, 2650
std::numeric_limits< unsigned char >, 2654
std::numeric_limits< unsigned int >, 2658
std::numeric_limits< unsigned long >, 2662
std::numeric_limits< unsigned long long >, 2666
std::numeric_limits< unsigned short >, 2670
std::numeric_limits< wchar_t >, 2674
is_sufficiently_aligned
Memory, 327
isalnum
std, 605
isalpha
std, 605
isblank
std, 605
iscntrl
std, 606
isctype
std::regex_traits< _Ch_type >, 2836
isdigit
std, 606
isgraph
std, 606
islower
std, 606
isprint
std, 606
ispunct
std, 606
isspace
std, 607
istream, 3617
I/O, 169
istream.tcc, 3342
istream_iterator
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2323, 2324
istream_type
std::istreambuf_iterator< _CharT, _Traits >, 2326
istreambuf_iterator
std::istreambuf_iterator< _CharT, _Traits >, 2327
istringstream
I/O, 169
isupper
std, 607
isxdigit
std, 607
iter_rvalue_reference_t
std, 588
iter_swap
Mutating, 28
iter_type
std::money_get< _CharT, _InIter >, 2472
std::money_put< _CharT, _OutIter >, 2476
std::num_get< _CharT, _InIter >, 2570
std::num_put< _CharT, _OutIter >, 2587
std::time_get< _CharT, _InIter >, 3011
std::time_put< _CharT, _OutIter >, 3034
iterator, 3618, 3619
std::set< _Key, _Compare, _Alloc >, 2883
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3117
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3146
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3171
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3195
Iterator Tags, 177
iterator.h, 3659
iterator.hpp, 3581
iterator_category
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1767
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1771
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1781
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1785
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2337
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_ite
Node, _Alloc >, 2341
std::back_insert_iterator< _Container >, 1099
std::front_insert_iterator< _Container >, 2193
std::insert_iterator< _Container >, 2283
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2323
std::istreambuf_iterator< _CharT, _Traits >, 2326
std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, 2331
std::ostream_iterator< _Tp, _CharT, _Traits >, 2690
std::ostreambuf_iterator< _CharT, _Traits >, 2694
std::raw_storage_iterator< _OutputIterator, _Tp >,

2806
iterator_concepts.h, 3343
iterator_fn_imps.hpp, 3564
Iterators, 171
back_inserter, 175
front_inserter, 175
inserter, 176
make_reverse_iterator, 176
operator==, 176
iterators_fn_imps.hpp, 3552
iword
std::basic_fstream< _CharT, _Traits >, 1154
std::basic_ifstream< _CharT, _Traits >, 1204
std::basic_ios< _CharT, _Traits >, 1239
std::basic_iostream< _CharT, _Traits >, 1269
std::basic_istream< _CharT, _Traits >, 1316
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1356
std::basic_ofstream< _CharT, _Traits >, 1391
std::basic_ostream< _CharT, _Traits >, 1424
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1457
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1732
std::ios_base, 2293

k
std::negative_binomial_distribution< _IntType >, 2554
key_comp
std::map< _Key, _Tp, _Compare, _Alloc >, 2429
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2522
std::multiset< _Key, _Compare, _Alloc >, 2544
std::set< _Key, _Compare, _Alloc >, 2896
key_compare
std::set< _Key, _Compare, _Alloc >, 2883
key_eq
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3135
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3162
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3186
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3211
key_equal
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3117
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3146
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3171
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3195
key_type
std::set< _Key, _Compare, _Alloc >, 2884
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3117
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3146
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3172
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3196
kill_dependency
Atomics, 116

L1_cache_size
__gnu_parallel::_Settings, 1032
L2_cache_size
__gnu_parallel::_Settings, 1033
laguerre
Mathematical Special Functions, 214
TR1 Mathematical Special Functions, 254
laguerref
Mathematical Special Functions, 215
laguerrel
Mathematical Special Functions, 215
lambda
std::exponential_distribution< _RealType >, 2150
latch, 3619
launch
Futures, 119
lcm
std::experimental, 706
left
std, 607
std::basic_fstream< _CharT, _Traits >, 1182
std::basic_ifstream< _CharT, _Traits >, 1225
std::basic_ios< _CharT, _Traits >, 1249
std::basic_iostream< _CharT, _Traits >, 1296
std::basic_istream< _CharT, _Traits >, 1336
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1376
std::basic_ofstream< _CharT, _Traits >, 1410
std::basic_ostream< _CharT, _Traits >, 1441
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1474
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1759
std::ios_base, 2300
left_child_next_sibling_heap_.hpp, 3567
left_child_next_sibling_heap_const_iterator_
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2337, 2338
left_child_next_sibling_heap_node_point_const_iterator_

__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator, Node, _Alloc >, 2342
legendre
Mathematical Special Functions, 215
TR1 Mathematical Special Functions, 254
legendref
Mathematical Special Functions, 216
legendrel
Mathematical Special Functions, 216
length
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 863
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1560
std::basic_string< _CharT, _Traits, _Alloc >, 1654
std::match_results< _Bi_iter, _Alloc >, 2443
std::regex_traits< _Ch_type >, 2836
std::sub_match< _BiIter >, 2985
lexicographical_compare
Sorting, 69
lexicographical_compare_3way
SGI, 159
lexicographical_compare_three_way
Sorting, 70
lfts_config.h, 3530
Library Fundamentals TS, 285
limits, 3619
linear_congruential_engine
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2351
linear_probe_fn_imp.hpp, 3564
list, 3621, 3622
std::list< _Tp, _Alloc >, 2362, 2363
List-Based, 148
list.tcc, 3344
list_partition
__gnu_parallel, 462
list_partition.h, 3659
list_update
__gnu_pbds::list_update< Key, Mapped, Eq_Fn, Update_Policy, _Alloc >, 2380
list_update_policy.hpp, 3583
load_factor
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3135
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3162
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3186
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3211
local_iterator
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3117
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3146
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3172
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3196
local_time_format
Time, 379
locale, 3622
std::locale, 2382–2384
locale_classes.h, 3345
locale_classes.tcc, 3345
locale_conv.h, 3345
locale_facets.h, 3346
locale_facets.tcc, 3347
locale_facets_nonio.h, 3348
locale_facets_nonio.tcc, 3349
localefwd.h, 3349
Locales, 177
has_facet, 178
use_facet, 178
lock
Mutexes, 122
log
Complex Numbers, 185
math.h, 3626
log10
Complex Numbers, 186
math.h, 3626
logic_error
std::logic_error, 2391
Logical operator traits, 292
lookup_classname
std::regex_traits< _Ch_type >, 2837
lookup_collatename
std::regex_traits< _Ch_type >, 2838
losertree.h, 3659
lower_bound
Binary Search, 87, 88
std::map< _Key, _Tp, _Compare, _Alloc >, 2429, 2430
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2522, 2523
std::multiset< _Key, _Compare, _Alloc >, 2544
std::set< _Key, _Compare, _Alloc >, 2896
lowest
std::numeric_limits< _Tp >, 2600
lu_counter_metadata.hpp, 3570
lu_map_.hpp, 3569

macros.h, 3508
_GLIBCXX_DEBUG_VERIFY_AT_F, 3511
__glibcxx_check_erase, 3509
__glibcxx_check_erase_after, 3509

__glibcxx_check_erase_range, 3509
__glibcxx_check_erase_range_after, 3509
__glibcxx_check_heap_pred, 3510
__glibcxx_check_insert, 3510
__glibcxx_check_insert_after, 3510
__glibcxx_check_insert_range, 3510
__glibcxx_check_insert_range_after, 3510
__glibcxx_check_partitioned_lower, 3511
__glibcxx_check_partitioned_lower_pred, 3511
__glibcxx_check_partitioned_upper_pred, 3511
__glibcxx_check_sorted_pred, 3511
make12
Time, 379
make24
Time, 379
make_array
Array creation functions, 287
make_boyer_moore_horspool_searcher
std::experimental, 706
make_boyer_moore_searcher
std::experimental, 706
make_default_searcher
std::experimental, 706
make_error_code
Diagnostics, 130
Futures, 120
make_error_condition
Diagnostics, 130
Futures, 120
make_exception_ptr
Exceptions, 135
make_heap
Heap, 92, 93
make_index_sequence
std, 588
make_integer_sequence
std, 588
make_ostream_joiner
std::experimental, 707
make_pair
Utilities, 309
make_reverse_iterator
Iterators, 176
make_shared
Pointer Abstractions, 348
make_signed_t
Metaprogramming, 360
make_tuple
Utilities, 309
make_unsigned_t
Metaprogramming, 360
malloc_allocator.h, 3537
map, 3622, 3623
std::map< _Key, _Tp, _Compare, _Alloc >, 2415–2417
map.h, 3511
mapped_type
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3117
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3146
mark_count
std::basic_regex< _Ch_type, _Rx_traits >, 1484
mask_array
Numeric Arrays, 223
mask_array.h, 3350
mask_based_range_hashing.hpp, 3565
match_any
std::regex_constants, 721
match_continuous
std::regex_constants, 721
match_default
std::regex_constants, 722
match_flag_type
std::regex_constants, 715
match_not_bol
std::regex_constants, 722
match_not_bow
std::regex_constants, 722
match_not_eol
std::regex_constants, 722
match_not_eow
std::regex_constants, 722
match_not_null
std::regex_constants, 722
match_prev_avail
std::regex_constants, 722
match_results
std::match_results< _Bi_iter, _Alloc >, 2440
math.h, 3623
abs, 3625
cos, 3625
cosh, 3625
exp, 3625
fabs, 3626
log, 3626
log10, 3626
pow, 3626
sin, 3626
sinh, 3626
sqrt, 3626
tan, 3626
tanh, 3627
Mathematical Special Functions, 191
airy_ai, 195
airy_aif, 195
airy_ail, 195

airy_bi, 195
airy_bif, 195
airy_bil, 196
assoc_laguerre, 196
assoc_laguerref, 197
assoc_laguerrel, 197
assoc_legendre, 197
assoc_legendref, 198
assoc_legendrel, 198
beta, 198
betaf, 199
betal, 199
comp_ellint_1, 199
comp_ellint_1f, 200
comp_ellint_1l, 200
comp_ellint_2, 200
comp_ellint_2f, 201
comp_ellint_2l, 201
comp_ellint_3, 201
comp_ellint_3f, 202
comp_ellint_3l, 202
conf_hyperg, 203
conf_hypergf, 203
conf_hypergl, 203
cyl_bessel_i, 203
cyl_bessel_if, 204
cyl_bessel_il, 204
cyl_bessel_j, 204
cyl_bessel_jf, 205
cyl_bessel_jl, 205
cyl_bessel_k, 205
cyl_bessel_kf, 206
cyl_bessel_kl, 206
cyl_neumann, 207
cyl_neumannf, 207
cyl_neumannl, 207
ellint_1, 208
ellint_1f, 208
ellint_1l, 208
ellint_2, 209
ellint_2f, 209
ellint_2l, 210
ellint_3, 210
ellint_3f, 211
ellint_3l, 211
expint, 211
expintf, 212
expintl, 212
hermite, 212
hermitef, 212
hermitel, 213
hyperg, 213
hypergf, 213
hypergl, 214
laguerre, 214
laguerref, 215
laguerrel, 215
legendre, 215
legendref, 216
legendrel, 216
riemann_zeta, 216
riemann_zetaf, 217
riemann_zetal, 217
sph_bessel, 217
sph_besself, 218
sph_bessell, 218
sph_legendre, 218
sph_legendref, 219
sph_legendrel, 219
sph_neumann, 219
sph_neumannf, 220
sph_neumannl, 220
max
__gnu_parallel, 462
Numeric Arrays, 227
Sorting, 70, 71
std::bernoulli_distribution, 1762
std::binomial_distribution< _IntType >, 1792
std::cauchy_distribution< _RealType >, 1817
std::chi_squared_distribution< _RealType >, 1843
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2085
std::discrete_distribution< _IntType >, 2089
std::exponential_distribution< _RealType >, 2150
std::extreme_value_distribution< _RealType >, 2154
std::fisher_f_distribution< _RealType >, 2164
std::gamma_distribution< _RealType >, 2210
std::geometric_distribution< _IntType >, 2214
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2274
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2351
std::lognormal_distribution< _RealType >, 2399
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2456
std::negative_binomial_distribution< _IntType >, 2554
std::normal_distribution< _RealType >, 2561
std::numeric_limits< _Tp >, 2600
std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2750
std::piecewise_constant_distribution< _RealType >, 2754
std::piecewise_linear_distribution< _RealType >, 2758
std::poisson_distribution< _IntType >, 2770

std::shuffle_order_engine< _RandomNumberEngine, __k >, 2918
std::student_t_distribution< _RealType >, 2980
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2989
std::uniform_int_distribution< _IntType >, 3093
std::uniform_real_distribution< _RealType >, 3096
std::weibull_distribution< _RealType >, 3270
max_blocks_per_chunk
std::pmr::pool_options, 2773
max_bucket_count
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3135
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3162
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3186
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3211
max_count
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2403
max_digits10
std::__numeric_limits_base, 809
std::numeric_limits< _Tp >, 2602
std::numeric_limits< bool >, 2606
std::numeric_limits< char >, 2610
std::numeric_limits< char16_t >, 2614
std::numeric_limits< char32_t >, 2618
std::numeric_limits< double >, 2622
std::numeric_limits< float >, 2626
std::numeric_limits< int >, 2630
std::numeric_limits< long >, 2634
std::numeric_limits< long double >, 2638
std::numeric_limits< long long >, 2642
std::numeric_limits< short >, 2646
std::numeric_limits< signed char >, 2650
std::numeric_limits< unsigned char >, 2654
std::numeric_limits< unsigned int >, 2658
std::numeric_limits< unsigned long >, 2662
std::numeric_limits< unsigned long long >, 2666
std::numeric_limits< unsigned short >, 2670
std::numeric_limits< wchar_t >, 2674
max_element
Sorting, 71, 72
max_element_minimal_n
__gnu_parallel::_Settings, 1033
max_exponent
std::__numeric_limits_base, 809
std::numeric_limits< _Tp >, 2602
std::numeric_limits< bool >, 2606
std::numeric_limits< char >, 2610
std::numeric_limits< char16_t >, 2614
std::numeric_limits< char32_t >, 2618
std::numeric_limits< double >, 2622
std::numeric_limits< float >, 2626
std::numeric_limits< int >, 2630
std::numeric_limits< long >, 2634
std::numeric_limits< long double >, 2638
std::numeric_limits< long long >, 2642
std::numeric_limits< short >, 2646
std::numeric_limits< signed char >, 2650
std::numeric_limits< unsigned char >, 2654
std::numeric_limits< unsigned int >, 2658
std::numeric_limits< unsigned long >, 2662
std::numeric_limits< unsigned long long >, 2666
std::numeric_limits< unsigned short >, 2670
std::numeric_limits< wchar_t >, 2674
max_exponent10
std::__numeric_limits_base, 809
std::numeric_limits< _Tp >, 2602
std::numeric_limits< bool >, 2606
std::numeric_limits< char >, 2610
std::numeric_limits< char16_t >, 2614
std::numeric_limits< char32_t >, 2618
std::numeric_limits< double >, 2622
std::numeric_limits< float >, 2626
std::numeric_limits< int >, 2630
std::numeric_limits< long >, 2634
std::numeric_limits< long double >, 2638
std::numeric_limits< long long >, 2642
std::numeric_limits< short >, 2646
std::numeric_limits< signed char >, 2650
std::numeric_limits< unsigned char >, 2654
std::numeric_limits< unsigned int >, 2658
std::numeric_limits< unsigned long >, 2662
std::numeric_limits< unsigned long long >, 2666
std::numeric_limits< unsigned short >, 2670
std::numeric_limits< wchar_t >, 2674
max_load_factor
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3135
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3162
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3186
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3211
max_size
__gnu_cxx::__alloc_traits< _Alloc, typename >, 754
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 863
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1560
std::allocator_traits< _Alloc >, 1059
std::allocator_traits< allocator< _Tp > >, 1064
std::allocator_traits< allocator< void > >, 1068

std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1073
std::basic_string< _CharT, _Traits, _Alloc >, 1654
std::deque< _Tp, _Alloc >, 2073
std::forward_list< _Tp, _Alloc >, 2182
std::list< _Tp, _Alloc >, 2371
std::map< _Key, _Tp, _Compare, _Alloc >, 2430
std::match_results< _Bi_iter, _Alloc >, 2443
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2523
std::multiset< _Key, _Compare, _Alloc >, 2545
std::set< _Key, _Compare, _Alloc >, 2897
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2103
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3136
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3163
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3186
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3212
std::vector< _Tp, _Alloc >, 3240
max_size_type.h, 3350
mdspan, 3627
mean
std::normal_distribution< _RealType >, 2561
std::poisson_distribution< _IntType >, 2770
mem_fn
Function Objects, 316
Memory, 325
align, 326
assume_aligned, 327
is_sufficiently_aligned, 327
uninitialized_copy, 327
uninitialized_copy_n, 328
uninitialized_default_construct, 328
uninitialized_default_construct_n, 329
uninitialized_fill, 329
uninitialized_fill_n, 329
uninitialized_move, 330
uninitialized_move_n, 330
uninitialized_value_construct, 331
uninitialized_value_construct_n, 331
memory, 3631–3633
memory_order
Atomics, 116
memory_resource, 3633, 3634
get_default_resource, 3634
null_memory_resource, 3634
set_default_resource, 3634
memory_resource.h, 3350
memoryfwd.h, 3351
merge
Sorting, 72, 73
std::forward_list< _Tp, _Alloc >, 2182
std::list< _Tp, _Alloc >, 2371
merge.h, 3660
merge_minimal_n
__gnu_parallel::_Settings, 1033
merge_oversampling
__gnu_parallel::_Settings, 1033
mersenne_twister_engine
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2456
message
std::error_category, 2140
std::error_code, 2142
std::error_condition, 2144
messages
std::locale, 2389
std::messages< _CharT >, 2460, 2461
messages_members.h, 3724
metadata_const_reference
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1767
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1771
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 957
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 960
metadata_reference
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2402
__gnu_pbds::lu_move_to_front_policy< _Alloc >, 2407
metadata_type
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1767
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1772
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 957
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 960
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2402
__gnu_pbds::lu_move_to_front_policy< _Alloc >, 2407
__gnu_pbds::sample_update_policy, 2868
Metaprogramming, 354
add_lvalue_reference_t, 359
add_pointer_t, 359

add_rvalue_reference_t, 359
aligned_storage_t, 359
alignment_value, 362
common_reference_t, 359
common_type_t, 359
conditional_t, 359
decay_t, 359
enable_if_t, 359
false_type, 359
is_corresponding_member, 361
is_layout_compatible_v, 362
is_pointer_interconvertible_base_of_v, 362
is_pointer_interconvertible_with_class, 361
make_signed_t, 360
make_unsigned_t, 360
remove_all_extents_t, 360
remove_extent_t, 360
remove_pointer_t, 360
remove_reference_t, 360
result_of_t, 360
swap, 361
true_type, 360
type, 360
underlying_type_t, 361
microseconds
Time, 374
milliseconds
Time, 374
min
__gnu_parallel, 463
Numeric Arrays, 227
Sorting, 74
std::bernoulli_distribution, 1762
std::binomial_distribution< _IntType >, 1792
std::cauchy_distribution< _RealType >, 1817
std::chi_squared_distribution< _RealType >, 1843
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2085
std::discrete_distribution< _IntType >, 2089
std::exponential_distribution< _RealType >, 2150
std::extreme_value_distribution< _RealType >, 2154
std::fisher_f_distribution< _RealType >, 2164
std::gamma_distribution< _RealType >, 2210
std::geometric_distribution< _IntType >, 2214
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2274
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2352
std::lognormal_distribution< _RealType >, 2399
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2456
std::negative_binomial_distribution< _IntType >, 2554
std::normal_distribution< _RealType >, 2561
std::numeric_limits< _Tp >, 2600
std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2750
std::piecewise_constant_distribution< _RealType >, 2754
std::piecewise_linear_distribution< _RealType >, 2758
std::poisson_distribution< _IntType >, 2770
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2918
std::student_t_distribution< _RealType >, 2980
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2989
std::uniform_int_distribution< _IntType >, 3093
std::uniform_real_distribution< _RealType >, 3096
std::weibull_distribution< _RealType >, 3270
min_element
Sorting, 75
min_element_minimal_n
__gnu_parallel::_Settings, 1033
min_exponent
std::__numeric_limits_base, 809
std::numeric_limits< _Tp >, 2602
std::numeric_limits< bool >, 2606
std::numeric_limits< char >, 2610
std::numeric_limits< char16_t >, 2614
std::numeric_limits< char32_t >, 2618
std::numeric_limits< double >, 2622
std::numeric_limits< float >, 2626
std::numeric_limits< int >, 2630
std::numeric_limits< long >, 2634
std::numeric_limits< long double >, 2638
std::numeric_limits< long long >, 2642
std::numeric_limits< short >, 2646
std::numeric_limits< signed char >, 2650
std::numeric_limits< unsigned char >, 2654
std::numeric_limits< unsigned int >, 2658
std::numeric_limits< unsigned long >, 2662
std::numeric_limits< unsigned long long >, 2666
std::numeric_limits< unsigned short >, 2670
std::numeric_limits< wchar_t >, 2674
min_exponent10
std::__numeric_limits_base, 809
std::numeric_limits< _Tp >, 2602
std::numeric_limits< bool >, 2606
std::numeric_limits< char >, 2610
std::numeric_limits< char16_t >, 2614
std::numeric_limits< char32_t >, 2618
std::numeric_limits< double >, 2622
std::numeric_limits< float >, 2626
std::numeric_limits< int >, 2630
std::numeric_limits< long >, 2634
std::numeric_limits< long double >, 2638

std::numeric_limits< long long >, 2642
std::numeric_limits< short >, 2646
std::numeric_limits< signed char >, 2650
std::numeric_limits< unsigned char >, 2654
std::numeric_limits< unsigned int >, 2658
std::numeric_limits< unsigned long >, 2662
std::numeric_limits< unsigned long long >, 2666
std::numeric_limits< unsigned short >, 2670
std::numeric_limits< wchar_t >, 2674
minmax
Sorting, 76
minmax_element
Sorting, 77
minstd_rand
Random Number Generators, 248
minstd_rand0
Random Number Generators, 248
minutes
Time, 374
mismatch
Non-Mutating, 57, 58
mod_based_range_hashing.hpp, 3565
modulus
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2354
mofunc_impl.h, 3351
monetary
std::locale, 2389
money_get
std::money_get< _CharT, _InIter >, 2472
money_put
std::money_put< _CharT, _OutIter >, 2477
moneypunct
std::moneypunct< _CharT, _Intl >, 2482, 2483
monostate.h, 3351
months
Time, 374
move
Mutating, 28
Utilities, 309
move.h, 3351
move_backward
Mutating, 29
move_if_noexcept
Utilities, 311
move_only_function
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2500, 2501
mt19937
Random Number Generators, 248
mt19937_64
Random Number Generators, 249
mt_allocator.h, 3537
multiline
std::regex_constants, 722
multimap
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2509–2511
multimap.h, 3512
multiplier
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2354
multiseq_partition
__gnu_parallel, 463
multiseq_selection
__gnu_parallel, 463
multiseq_selection.h, 3660
multiset
std::multiset< _Key, _Compare, _Alloc >, 2533–2536
multiset.h, 3513
multiway_merge
__gnu_parallel, 464
multiway_merge.h, 3661
_GLIBCXX_PARALLEL_LENGTH, 3664
multiway_merge_3_variant
__gnu_parallel, 465
multiway_merge_4_variant
__gnu_parallel, 466
multiway_merge_exact_splitting
__gnu_parallel, 467
multiway_merge_loser_tree
__gnu_parallel, 467
multiway_merge_loser_tree_sentinel
__gnu_parallel, 467
multiway_merge_loser_tree_unguarded
__gnu_parallel, 468
multiway_merge_minimal_k
__gnu_parallel::_Settings, 1033
multiway_merge_minimal_n
__gnu_parallel::_Settings, 1033
multiway_merge_oversampling
__gnu_parallel::_Settings, 1033
multiway_merge_sampling_splitting
__gnu_parallel, 469
multiway_merge_sentinels
__gnu_parallel, 469
multiway_mergesort.h, 3664
Mutating, 21
copy, 23
copy_backward, 24
copy_if, 24
copy_n, 25
fill, 25
fill_n, 26
generate, 26
generate_n, 27

is_partitioned, 27
iter_swap, 28
move, 28
move_backward, 29
partition, 29
partition_copy, 30
partition_point, 30
random_shuffle, 31
remove, 32
remove_copy, 32
remove_copy_if, 33
remove_if, 33
replace, 34
replace_copy_if, 34
replace_if, 35
reverse, 35
reverse_copy, 36
rotate, 36
rotate_copy, 37
shuffle, 38
stable_partition, 38
swap_ranges, 39
transform, 39, 40
unique, 40, 41
unique_copy, 41, 42
mutex, 3635
Mutexes, 121
adopt_lock, 124
call_once, 122
defer_lock, 124
lock, 122
try_lock, 123
try_to_lock, 124

name
__gnu_debug::type_info, 3082
std::error_category, 2140
std::locale, 2386
std::type_info, 3083
nanoseconds
Time, 375
narrow
std::__ctype_abstract_base< _CharT >, 775, 776
std::basic_fstream< _CharT, _Traits >, 1154
std::basic_ifstream< _CharT, _Traits >, 1205
std::basic_ios< _CharT, _Traits >, 1240
std::basic_iostream< _CharT, _Traits >, 1270
std::basic_istream< _CharT, _Traits >, 1316
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1357
std::basic_ofstream< _CharT, _Traits >, 1392
std::basic_ostream< _CharT, _Traits >, 1424
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1457
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1732
std::ctype< _CharT >, 1952, 1953
std::ctype< char >, 1975–1977
std::ctype< wchar_t >, 1997
std::ctype_byname< _CharT >, 2011, 2012
std::ctype_byname< char >, 2031, 2032
native_handle
std::thread, 3001
neg_format
std::moneypunct< _CharT, _Intl >, 2487
std::moneypunct_byname< _CharT, _Intl >, 2494
negative_sign
std::moneypunct< _CharT, _Intl >, 2487
std::moneypunct_byname< _CharT, _Intl >, 2495
Negators, 324
not1, 325
not2, 325
nested_exception
std::nested_exception, 2557
nested_exception.h, 3352
nested_ptr
std::nested_exception, 2557
new, 3729
operator new, 3730
new_allocator.h, 3353
new_delete_resource
Polymorphic memory resources, 354
new_except.h, 3353
new_handler
std, 588
new_throw.h, 3354
next_permutation
Sorting, 78
noboolalpha
std, 607
node.hpp, 3568
node_begin
__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2700
__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >, 2740
__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2813
__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2931
node_const_iterator
__gnu_pbds::detail::bin_search_tree_traits< Key, Mapped, Cmp_Fn, Node_Update, Node, _Alloc >, 1774
__gnu_pbds::detail::bin_search_tree_traits< Key, null_type, Cmp_Fn, Node_Update, Node, _Alloc >, 1774
__gnu_pbds::detail::tree_traits< Key, Mapped,

Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, 3049
__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, 3051
__gnu_pbds::detail::tree_traits< Key, Mapped, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, 3052
__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, ov_tree_tag, _Alloc >, 3053
__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, rb_tree_tag, _Alloc >, 3054
__gnu_pbds::detail::tree_traits< Key, null_type, Cmp_Fn, Node_Update, splay_tree_tag, _Alloc >, 3055
__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3070
__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3071
node_end
__gnu_pbds::detail::ov_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2700
__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >, 2741
__gnu_pbds::detail::rb_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2813
__gnu_pbds::detail::splay_tree_map< Key, Mapped, Cmp_Fn, Node_And_It_Traits, _Alloc >, 2931
node_handle.h, 3354
node_iterators.hpp, 3539
node_metadata_selector.hpp, 3578
node_type
__gnu_pbds::detail::pat_trie_base, 2738
__gnu_pbds::detail::pat_trie_map< Key, Mapped, Node_And_It_Traits, _Alloc >, 2740
node_update
__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3070
__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3071
Non-Mutating, 43
adjacent_find, 45
all_of, 45
any_of, 46
count, 46
count_if, 47
equal, 47–49
find, 49
find_end, 50, 51
find_first_of, 51, 52
find_if, 53
find_if_not, 53
for_each, 54
for_each_n, 54
is_permutation, 55, 56
mismatch, 57, 58
none_of, 59
search, 59, 60
search_n, 61
none
std::bitset< _Nb >, 1808
std::locale, 2389
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2103
none_of
Non-Mutating, 59
norm
Complex Numbers, 186
std::tr1, 731
Normal Distributions, 240
operator<<, 240
operator>>, 241
normal_distribution
std::normal_distribution< _RealType >, 2561
noshowbase
std, 607
noshowpoint
std, 607
noshowpos
std, 608
noskipws
std, 608
nosubs
std::regex_constants, 723
not1
Negators, 325
not2
Negators, 325
not_fn
std::experimental, 707
notify_cleared
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1820
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2255
__gnu_pbds::sample_resize_policy, 2860
__gnu_pbds::sample_resize_trigger, 2863
notify_erase_search_collision
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1820
__gnu_pbds::sample_resize_policy, 2860
__gnu_pbds::sample_resize_trigger, 2863
notify_erase_search_end

__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1820
__gnu_pbds::sample_resize_policy, 2860
__gnu_pbds::sample_resize_trigger, 2863
notify_erase_search_start
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1820
__gnu_pbds::sample_resize_policy, 2860
__gnu_pbds::sample_resize_trigger, 2863
notify_erased
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1821
__gnu_pbds::sample_resize_policy, 2860
__gnu_pbds::sample_resize_trigger, 2863
notify_externally_resized
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1821
__gnu_pbds::sample_resize_trigger, 2863
notify_find_search_collision
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1821
__gnu_pbds::sample_resize_policy, 2860
__gnu_pbds::sample_resize_trigger, 2863
notify_find_search_end
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1821
__gnu_pbds::sample_resize_policy, 2861
__gnu_pbds::sample_resize_trigger, 2863
notify_find_search_start
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1821
__gnu_pbds::sample_resize_policy, 2861
__gnu_pbds::sample_resize_trigger, 2863
notify_insert_search_collision
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1821
__gnu_pbds::sample_resize_policy, 2861
__gnu_pbds::sample_resize_trigger, 2864
notify_insert_search_end
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1821
__gnu_pbds::sample_resize_policy, 2861
__gnu_pbds::sample_resize_trigger, 2864
notify_insert_search_start
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1822
__gnu_pbds::sample_resize_policy, 2861
__gnu_pbds::sample_resize_trigger, 2864
notify_inserted
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1822
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2255
__gnu_pbds::sample_resize_policy, 2861
__gnu_pbds::sample_resize_trigger, 2864
notify_resized
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1822
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2255
__gnu_pbds::sample_range_hashing, 2857
__gnu_pbds::sample_ranged_hash_fn, 2858
__gnu_pbds::sample_resize_policy, 2861
__gnu_pbds::sample_resize_trigger, 2864
__gnu_pbds::sample_resize_trigger, 2864
notifybuf
std, 608
nouppercase
std, 608
npos
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 879
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1581
std::basic_string< _CharT, _Traits, _Alloc >, 1686
nth_element
Sorting, 79
nth_element_minimal_n
__gnu_parallel::_Settings, 1033
null_memory_resource
memory_resource, 3634
null_node_metadata.hpp, 3557
nullopt
Optional values, 293
num_blocks
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2103
num_children
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 957
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 960
num_get
std::num_get< _CharT, _InIter >, 2570
num_put
std::num_put< _CharT, _OutIter >, 2587
numbers, 3635
numeric, 3635–3637
std::locale, 2389
Numeric Arrays, 220
~gslice, 225
apply, 225, 226
cshift, 226
gslice, 222
gslice_array, 223
indirect_array, 223
mask_array, 223
max, 227

min, 227
operator=, 227–230
operator[], 231–234
resize, 234
shift, 235
size, 235
slice, 223, 224
slice_array, 224
start, 235, 236
stride, 236
sum, 236
swap, 236
valarray, 224, 225
numeric_traits.h, 3530
numericfwd.h, 3664
Numerics, 180
numpunct
std::numpunct< _CharT >, 2677

oct
std, 608
std::basic_fstream< _CharT, _Traits >, 1182
std::basic_ifstream< _CharT, _Traits >, 1225
std::basic_ios< _CharT, _Traits >, 1249
std::basic_iostream< _CharT, _Traits >, 1296
std::basic_istream< _CharT, _Traits >, 1336
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1376
std::basic_ofstream< _CharT, _Traits >, 1410
std::basic_ostream< _CharT, _Traits >, 1441
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1475
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1759
std::ios_base, 2300
off_type
std::basic_ios< _CharT, _Traits >, 1233
std::basic_streambuf< _CharT, _Traits >, 1489
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3253
ofstream
I/O, 169
omp_loop.h, 3666
omp_loop_static.h, 3666
once_flag
std::once_flag, 2687
open
__gnu_cxx::enc_filebuf< _CharT >, 2117
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2944, 2945
std::basic_filebuf< _CharT, _Traits >, 1119, 1120
std::basic_fstream< _CharT, _Traits >, 1155
std::basic_ifstream< _CharT, _Traits >, 1205, 1206
std::basic_ofstream< _CharT, _Traits >, 1392, 1393
openmode
std::basic_fstream< _CharT, _Traits >, 1141
std::basic_ifstream< _CharT, _Traits >, 1192
std::basic_ios< _CharT, _Traits >, 1233
std::basic_iostream< _CharT, _Traits >, 1258
std::basic_istream< _CharT, _Traits >, 1305
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1345
std::basic_ofstream< _CharT, _Traits >, 1384
std::basic_ostream< _CharT, _Traits >, 1418
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1450
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1720
std::ios_base, 2291
operator _Iterator
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 989
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1003
operator _RAIter
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 908
operator bool
std::basic_fstream< _CharT, _Traits >, 1156
std::basic_ifstream< _CharT, _Traits >, 1206
std::basic_ios< _CharT, _Traits >, 1240
std::basic_iostream< _CharT, _Traits >, 1270
std::basic_istream< _CharT, _Traits >, 1317
std::basic_istream< _CharT, _Traits >::sentry, 2875
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1357
std::basic_ofstream< _CharT, _Traits >, 1393
std::basic_ostream< _CharT, _Traits >, 1425
std::basic_ostream< _CharT, _Traits >::sentry, 2877
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1458
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1732
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1941
std::error_code, 2142
std::error_condition, 2144
std::function< _Res(_ArgTypes...)>, 2197
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2502
std::shared_ptr< _Tp >, 2914
std::tr2::bool_set, 1814
std::unique_ptr< _Tp, _Dp >, 3102
std::unique_ptr< _Tp[], _Dp >, 3107
operator new
new, 3730
operator streamoff
std::fpos< _StateT >, 2189
operator string_type

std::sub_match< _BiIter >, 2985
operator!
std::basic_fstream< _CharT, _Traits >, 1156
std::basic_ifstream< _CharT, _Traits >, 1206
std::basic_ios< _CharT, _Traits >, 1240
std::basic_iostream< _CharT, _Traits >, 1270
std::basic_istream< _CharT, _Traits >, 1317
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1357
std::basic_ofstream< _CharT, _Traits >, 1393
std::basic_ostream< _CharT, _Traits >, 1425
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1458
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1733
std::valarray< _Tp >, 3219
operator!=
__gnu_cxx, 404
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1768
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1772
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1782
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1786
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2338
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >, 2342
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 957
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 960
Dynamic Bitset., 140
std, 608, 609
std::rel_ops, 723
operator<
__gnu_cxx, 407
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 908
Pointer Abstractions, 348, 349
std, 611–613
Time, 383
operator<<
Bernoulli Distributions, 238, 239
Complex Numbers, 188
Diagnostics, 130
Dynamic Bitset., 141
Normal Distributions, 240
Pointer Abstractions, 349
Poisson Distributions, 242, 243
Random Number Generators, 249
Regular Expressions, 260
std, 613–619
std::__detail, 661
std::basic_fstream< _CharT, _Traits >, 1156–1161
std::basic_iostream< _CharT, _Traits >, 1270–1275
std::basic_ofstream< _CharT, _Traits >, 1394–1399
std::basic_ostream< _CharT, _Traits >, 1425–1430
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1458–1463
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1733–1738
std::binomial_distribution< _IntType >, 1793
std::bitset< _Nb >, 1808
std::chi_squared_distribution< _RealType >, 1844
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2086
std::discrete_distribution< _IntType >, 2090
std::filesystem::path, 2748
std::fisher_f_distribution< _RealType >, 2165
std::gamma_distribution< _RealType >, 2211
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2353
std::lognormal_distribution< _RealType >, 2400
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2457
std::negative_binomial_distribution< _IntType >, 2555
std::normal_distribution< _RealType >, 2562
std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2751
std::piecewise_constant_distribution< _RealType >, 2755
std::piecewise_linear_distribution< _RealType >, 2759
std::poisson_distribution< _IntType >, 2771
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2919
std::student_t_distribution< _RealType >, 2981
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2990
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2104
std::unique_ptr< _Tp[], _Dp >, 3109
Time, 383
Uniform Distributions, 245, 246
operator<<=
std::bitset< _Nb >, 1808
std::gslice_array< _Tp >, 2235
std::indirect_array< _Tp >, 2277
std::mask_array< _Tp >, 2435
std::slice_array< _Tp >, 2924
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2104
std::valarray< _Tp >, 3221

operator<=
__gnu_cxx, 408, 409
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 908
Dynamic Bitset., 141
Pointer Abstractions, 349
std, 619, 620
std::rel_ops, 724
Time, 383
operator<=>
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 919
Diagnostics, 131
Regular Expressions, 260, 261
std, 620–623
std::error_category, 2141
std::filesystem::path, 2748
Time, 384
Utilities, 311
operator>
__gnu_cxx, 410, 411
Dynamic Bitset., 141
Pointer Abstractions, 350
std, 628, 629
std::rel_ops, 724
Time, 384
operator>>
Bernoulli Distributions, 239
Complex Numbers, 188
Dynamic Bitset., 142
Normal Distributions, 241
Poisson Distributions, 243, 244
std, 630–635
std::__detail, 661
std::basic_fstream< _CharT, _Traits >, 1161–1166
std::basic_ifstream< _CharT, _Traits >, 1207–1212
std::basic_iostream< _CharT, _Traits >, 1276–1281
std::basic_istream< _CharT, _Traits >, 1317–1322
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1358–1363
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1738–1743
std::binomial_distribution< _IntType >, 1794
std::bitset< _Nb >, 1809
std::chi_squared_distribution< _RealType >, 1844
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2087
std::discrete_distribution< _IntType >, 2090
std::filesystem::path, 2748
std::fisher_f_distribution< _RealType >, 2165
std::gamma_distribution< _RealType >, 2212
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2275
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2353
std::lognormal_distribution< _RealType >, 2400
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2458
std::negative_binomial_distribution< _IntType >, 2556
std::normal_distribution< _RealType >, 2563
std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2752
std::piecewise_constant_distribution< _RealType >, 2755
std::piecewise_linear_distribution< _RealType >, 2759
std::poisson_distribution< _IntType >, 2771
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2920
std::student_t_distribution< _RealType >, 2982
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2990
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2105
Uniform Distributions, 246, 247
operator>>=
std::bitset< _Nb >, 1809
std::gslice_array< _Tp >, 2235
std::indirect_array< _Tp >, 2278
std::mask_array< _Tp >, 2435
std::slice_array< _Tp >, 2924
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2105
std::valarray< _Tp >, 3221
operator>=
__gnu_cxx, 411, 412
Dynamic Bitset., 141
Pointer Abstractions, 351
std, 629, 630
std::rel_ops, 724
Time, 384
operator()
__gnu_cxx::subtractive_rng, 2992
__gnu_parallel::_Nothing, 963
__gnu_parallel::_RandomNumber, 974
__gnu_parallel::__accumulate_selector< _It >, 745
__gnu_parallel::__adjacent_find_selector, 748
__gnu_parallel::__count_if_selector< _It, _Diff >, 764
__gnu_parallel::__count_selector< _It, _Diff >, 766
__gnu_parallel::__fill_selector< _It >, 783
__gnu_parallel::__find_first_of_selector< _FIterator >, 784
__gnu_parallel::__find_if_selector, 786
__gnu_parallel::__for_each_selector< _It >, 787
__gnu_parallel::__generate_selector< _It >, 789
__gnu_parallel::__identity_selector< _It >, 793

__gnu_parallel::__inner_product_selector< _It, _It2, _Tp >, 795
__gnu_parallel::__mismatch_selector, 799
__gnu_parallel::__replace_if_selector< _It, _Op, _Tp >, 820
__gnu_parallel::__replace_selector< _It, _Tp >, 822
__gnu_parallel::__transform1_selector< _It >, 823
__gnu_parallel::__transform2_selector< _It >, 825
__gnu_pbds::direct_mask_range_hashing< Size_Type >, 2079
__gnu_pbds::direct_mod_range_hashing< Size_Type >, 2080
__gnu_pbds::linear_probe_fn< Size_Type >, 2355
__gnu_pbds::lu_counter_policy< Max_Count, _Alloc >, 2403
__gnu_pbds::lu_move_to_front_policy< _Alloc >, 2407
__gnu_pbds::quadratic_probe_fn< Size_Type >, 2785
__gnu_pbds::sample_probe_fn, 2856
__gnu_pbds::sample_range_hashing, 2857
__gnu_pbds::sample_ranged_hash_fn, 2858
__gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 2867
__gnu_pbds::sample_update_policy, 2868
__gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >, 3047
__gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3061
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3065
std::bernoulli_distribution, 1762
std::binomial_distribution< _IntType >, 1792
std::cauchy_distribution< _RealType >, 1817
std::chi_squared_distribution< _RealType >, 1843
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1941
std::default_delete< _Tp >, 2047
std::default_delete< _Tp[]>, 2048
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2086
std::discrete_distribution< _IntType >, 2089
std::exponential_distribution< _RealType >, 2150
std::extreme_value_distribution< _RealType >, 2154
std::fisher_f_distribution< _RealType >, 2164
std::function< _Res(_ArgTypes...)>, 2197
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2202
std::gamma_distribution< _RealType >, 2210, 2211
std::geometric_distribution< _IntType >, 2214
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2274
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2352
std::locale, 2386
std::lognormal_distribution< _RealType >, 2399
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2502
std::negative_binomial_distribution< _IntType >, 2554
std::normal_distribution< _RealType >, 2561
std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2751
std::piecewise_constant_distribution< _RealType >, 2754
std::piecewise_linear_distribution< _RealType >, 2758
std::plus< _Tp >, 2761
std::poisson_distribution< _IntType >, 2770
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2918
std::student_t_distribution< _RealType >, 2980
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2989
std::uniform_int_distribution< _IntType >, 3093
std::uniform_real_distribution< _RealType >, 3096
std::weibull_distribution< _RealType >, 3270
operator+
__gnu_cxx, 405, 406
Complex Numbers, 186, 187
std, 609–611
std::fpos< _StateT >, 2189
std::reverse_iterator< _Iterator >, 2847
std::valarray< _Tp >, 3220
Time, 381, 382
operator++
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 989, 990
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1004
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 908
std::back_insert_iterator< _Container >, 1099
std::front_insert_iterator< _Container >, 2193, 2194
std::insert_iterator< _Container >, 2283
std::istreambuf_iterator< _CharT, _Traits >, 2327
std::ostreambuf_iterator< _CharT, _Traits >, 2695
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2829
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2833
std::reverse_iterator< _Iterator >, 2847
operator+=
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 863, 864
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1560
std::basic_string< _CharT, _Traits, _Alloc >, 1654–

1657
std::complex< _Tp >, 1905
std::fpos< _StateT >, 2189
std::gslice_array< _Tp >, 2235
std::indirect_array< _Tp >, 2277
std::mask_array< _Tp >, 2435
std::reverse_iterator< _Iterator >, 2848
std::slice_array< _Tp >, 2924
std::valarray< _Tp >, 3220
operator-
Complex Numbers, 187
Dynamic Bitset., 140
std::fpos< _StateT >, 2189
std::reverse_iterator< _Iterator >, 2848
std::valarray< _Tp >, 3220
Time, 382
operator->
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 990
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1004
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1783
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1786
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2338
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >, 2342
std::auto_ptr< _Tp >, 1095
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2829
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2833
std::reverse_iterator< _Iterator >, 2849
std::unique_ptr< _Tp, _Dp >, 3102
std::unique_ptr< _Tp[], _Dp >, 3108
operator--
std::reverse_iterator< _Iterator >, 2848
operator-=
std::complex< _Tp >, 1905
std::fpos< _StateT >, 2189
std::gslice_array< _Tp >, 2235
std::indirect_array< _Tp >, 2277
std::mask_array< _Tp >, 2435
std::reverse_iterator< _Iterator >, 2848
std::slice_array< _Tp >, 2924
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2104
std::valarray< _Tp >, 3220, 3221
operator/
Complex Numbers, 187, 188
std::filesystem::path, 2748
Time, 382, 383
operator/=
std::complex< _Tp >, 1906
std::gslice_array< _Tp >, 2235
std::indirect_array< _Tp >, 2277
std::mask_array< _Tp >, 2435
std::slice_array< _Tp >, 2924
std::valarray< _Tp >, 3221
operator=
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 865, 866
__gnu_cxx::pair< _T1, _T2 >, 2715, 2716
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 990
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1004
Numeric Arrays, 227–230
std::auto_ptr< _Tp >, 1095, 1096
std::back_insert_iterator< _Container >, 1100
std::basic_regex< _Ch_type, _Rx_traits >, 1484, 1485
std::basic_string< _CharT, _Traits, _Alloc >, 1657–1659
std::complex< _Tp >, 1906
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1941
std::deque< _Tp, _Alloc >, 2073, 2074
std::experimental::fundamentals_v1::any, 1077, 1078
std::forward_list< _Tp, _Alloc >, 2182, 2183
std::front_insert_iterator< _Container >, 2194
std::function< _Res(_ArgTypes...)>, 2197–2199
std::insert_iterator< _Container >, 2284
std::list< _Tp, _Alloc >, 2372
std::locale, 2386
std::map< _Key, _Tp, _Compare, _Alloc >, 2430, 2431
std::match_results< _Bi_iter, _Alloc >, 2443
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2502
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2523, 2524
std::multiset< _Key, _Compare, _Alloc >, 2545
std::once_flag, 2687
std::ostream_iterator< _Tp, _CharT, _Traits >, 2692
std::ostreambuf_iterator< _CharT, _Traits >, 2696
std::pair< _T1, _T2 >, 2720, 2721
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2829
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2834
std::set< _Key, _Compare, _Alloc >, 2897
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2105
std::unique_ptr< _Tp, _Dp >, 3102, 3103
std::unique_ptr< _Tp[], _Dp >, 3108
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3136

std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3163
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3187
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3212
std::vector< _Tp, _Alloc >, 3240, 3241
operator==
__gnu_cxx, 409, 410
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 919
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1768
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1773
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1783
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1786
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_, Node, _Alloc >, 2338
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >, 2342
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 958
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 961
Allocators, 333
Complex Numbers, 188
Diagnostics, 131, 132
Iterators, 176
Pointer Abstractions, 349, 350
Regular Expressions, 261–263
std, 623–628
std::__fwdlist::_Iterator< _Const, _Ptr >, 914
std::_Fwd_list_const_iterator< _Tp >, 904
std::_Fwd_list_iterator< _Tp >, 905
std::allocator< void >, 1052
std::bernoulli_distribution, 1763
std::binomial_distribution< _IntType >, 1794
std::bitset< _Nb >, 1809
std::cauchy_distribution< _RealType >, 1818
std::chi_squared_distribution< _RealType >, 1844
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1942
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2087
std::discrete_distribution< _IntType >, 2090
std::error_category, 2141
std::exponential_distribution< _RealType >, 2151
std::extreme_value_distribution< _RealType >, 2155
std::filesystem::path, 2748
std::fisher_f_distribution< _RealType >, 2165
std::gamma_distribution< _RealType >, 2212
std::geometric_distribution< _IntType >, 2215
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2275
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2324
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2353
std::locale, 2387
std::lognormal_distribution< _RealType >, 2400
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2457
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2503
std::negative_binomial_distribution< _IntType >, 2556
std::normal_distribution< _RealType >, 2563
std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2752
std::piecewise_constant_distribution< _RealType >, 2755
std::piecewise_linear_distribution< _RealType >, 2759
std::poisson_distribution< _IntType >, 2771
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2829
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2834
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2919
std::slice, 2922
std::student_t_distribution< _RealType >, 2981
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2990
std::uniform_int_distribution< _IntType >, 3094
std::uniform_real_distribution< _RealType >, 3097
std::weibull_distribution< _RealType >, 3271
Time, 384
Utilities, 311, 312
operator%
Time, 381
operator%=
std::gslice_array< _Tp >, 2234
std::indirect_array< _Tp >, 2277
std::mask_array< _Tp >, 2435
std::slice_array< _Tp >, 2924
std::valarray< _Tp >, 3219
operator&
Dynamic Bitset., 140
std, 609
std::regex_constants, 717
operator&=

std::bitset< _Nb >, 1808
std::gslice_array< _Tp >, 2234
std::indirect_array< _Tp >, 2277
std::mask_array< _Tp >, 2435
std::regex_constants, 717
std::slice_array< _Tp >, 2924
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2103, 2104
std::valarray< _Tp >, 3219
operator[]
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 866
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1560, 1561
Numeric Arrays, 231–234
std::basic_string< _CharT, _Traits, _Alloc >, 1660, 1661
std::bitset< _Nb >, 1809
std::deque< _Tp, _Alloc >, 2074
std::map< _Key, _Tp, _Compare, _Alloc >, 2431
std::match_results< _Bi_iter, _Alloc >, 2443
std::reverse_iterator< _Iterator >, 2849
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2105, 2106
std::unique_ptr< _Tp[], _Dp >, 3108
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3136, 3137
std::vector< _Tp, _Alloc >, 3241
operator*
__gnu_debug::_Safe_iterator< _Iterator, _Sequence, _Category >, 989
__gnu_debug::_Safe_local_iterator< _Iterator, _UContainer >, 1003
__gnu_parallel::_GuardedIterator< _RAIter, _Compare >, 908
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1768
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1773
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1783
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1786
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2338
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >, 2342
__gnu_pbds::detail::ov_tree_node_it_< Value_Type, Metadata_Type, _Alloc >, 2704
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 958
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 961
Complex Numbers, 186
std::auto_ptr< _Tp >, 1095
std::back_insert_iterator< _Container >, 1099
std::front_insert_iterator< _Container >, 2193
std::insert_iterator< _Container >, 2283
std::istreambuf_iterator< _CharT, _Traits >, 2327
std::ostreambuf_iterator< _CharT, _Traits >, 2695
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2829
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2833
std::reverse_iterator< _Iterator >, 2847
std::unique_ptr< _Tp, _Dp >, 3102
std::unique_ptr< _Tp[], _Dp >, 3107
Time, 381
operator*=
std::complex< _Tp >, 1905
std::gslice_array< _Tp >, 2235
std::indirect_array< _Tp >, 2277
std::mask_array< _Tp >, 2435
std::slice_array< _Tp >, 2924
std::valarray< _Tp >, 3220
operator~
std::bitset< _Nb >, 1810
std::regex_constants, 719
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2106
std::valarray< _Tp >, 3222
operator^
Dynamic Bitset., 142
std, 635
std::regex_constants, 718
operator^=
std::bitset< _Nb >, 1810
std::gslice_array< _Tp >, 2235
std::indirect_array< _Tp >, 2278
std::mask_array< _Tp >, 2436
std::regex_constants, 718
std::slice_array< _Tp >, 2925
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2106
std::valarray< _Tp >, 3222
operator|
Dynamic Bitset., 142
std, 635
std::regex_constants, 718
operator|=
std::bitset< _Nb >, 1810
std::gslice_array< _Tp >, 2235
std::indirect_array< _Tp >, 2278
std::mask_array< _Tp >, 2436
std::regex_constants, 719
std::slice_array< _Tp >, 2925
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2106
std::valarray< _Tp >, 3222

opt_random.h, 3724
optimize
    std::regex_constants, 723
optional, 3639, 3640
Optional values, 292
    in_place, 293
    nullopt, 293
optional_ref.h, 3354
order_of_key
    __gnu_pbds::tree_order_statistics_node_update< Node_CItr, Node_Itr, Cmp_Fn, _Alloc >, 3047
    __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3061
order_of_prefix
    __gnu_pbds::trie_order_statistics_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3061
order_preserving
    __gnu_pbds::container_traits< Cntnr >, 1933
order_statistics_imp.hpp, 3578, 3579
os_defines.h, 3724
ostream, 3640
    I/O, 169
ostream.h, 3354
ostream.tcc, 3356
ostream_insert.h, 3356
ostream_iterator
    std::ostream_iterator< _Tp, _CharT, _Traits >, 2691, 2692
ostream_print.h, 3356
ostream_type
    std::ostream_iterator< _Tp, _CharT, _Traits >, 2690
    std::ostreambuf_iterator< _CharT, _Traits >, 2694
ostreambuf_iterator
    std::ostreambuf_iterator< _CharT, _Traits >, 2695
ostringstream
    I/O, 170
out
    std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 762
    std::basic_fstream< _CharT, _Traits >, 1182
    std::basic_ifstream< _CharT, _Traits >, 1225
    std::basic_ios< _CharT, _Traits >, 1250
    std::basic_iostream< _CharT, _Traits >, 1297
    std::basic_istream< _CharT, _Traits >, 1336
    std::basic_istringstream< _CharT, _Traits, _Alloc >, 1377
    std::basic_ofstream< _CharT, _Traits >, 1410
    std::basic_ostream< _CharT, _Traits >, 1442
    std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1475
    std::basic_stringstream< _CharT, _Traits, _Alloc >, 1759
    std::codecvt< _InternT, _ExternT, _StateT >, 1848
    std::codecvt< _InternT, _ExternT, encoding_state >, 1855
    std::codecvt< char, char, mbstate_t >, 1862, 1863
    std::codecvt< char16_t, char, mbstate_t >, 1870
    std::codecvt< char32_t, char, mbstate_t >, 1877, 1878
    std::codecvt< wchar_t, char, mbstate_t >, 1885
    std::codecvt_byname< _InternT, _ExternT, _StateT >, 1892
    std::ios_base, 2300
out_ptr
    std, 636
out_ptr.h, 3356
ov_tree_map_.hpp, 3570
overflow
    __gnu_cxx::enc_filebuf< _CharT >, 2118
    __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2946
    __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2963
    std::basic_filebuf< _CharT, _Traits >, 1120
    std::basic_streambuf< _CharT, _Traits >, 1493
    std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1700
    std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3256
owner_before
    std::shared_ptr< _Tp >, 2914

p
    std::bernoulli_distribution, 1762
    std::binomial_distribution< _IntType >, 1793
    std::geometric_distribution< _IntType >, 2214
    std::negative_binomial_distribution< _IntType >, 2555
pair
    __gnu_cxx::pair< _T1, _T2 >, 2713
    std::pair< _T1, _T2 >, 2719
    Utilities, 304
pairing_heap_.hpp, 3571
par_loop.h, 3667
parallel.h, 3667
parallel_balanced
    __gnu_parallel, 431
parallel_multiway_merge
    __gnu_parallel, 471
parallel_omp_loop
    __gnu_parallel, 431
parallel_omp_loop_static
    __gnu_parallel, 431
parallel_sort_mwms
    __gnu_parallel, 471
parallel_sort_mwms_pu
    __gnu_parallel, 472
parallel_tag
    __gnu_parallel::parallel_tag, 2726
parallel_taskqueue

__gnu_parallel, 431
parallel_unbalanced
__gnu_parallel, 431
Parallelism TS, 300
parallelism_v2::simd_abi::deduce< _Tp, _Np,... >, 2045
param
std::bernoulli_distribution, 1762
std::binomial_distribution< _IntType >, 1793
std::cauchy_distribution< _RealType >, 1818
std::chi_squared_distribution< _RealType >, 1843
std::discrete_distribution< _IntType >, 2089
std::exponential_distribution< _RealType >, 2150, 2151
std::extreme_value_distribution< _RealType >, 2154
std::fisher_f_distribution< _RealType >, 2164
std::gamma_distribution< _RealType >, 2211
std::geometric_distribution< _IntType >, 2214
std::lognormal_distribution< _RealType >, 2399
std::negative_binomial_distribution< _IntType >, 2555
std::normal_distribution< _RealType >, 2562
std::piecewise_constant_distribution< _RealType >, 2754, 2755
std::piecewise_linear_distribution< _RealType >, 2758
std::poisson_distribution< _IntType >, 2770
std::student_t_distribution< _RealType >, 2981
std::uniform_int_distribution< _IntType >, 3094
std::uniform_real_distribution< _RealType >, 3096
std::weibull_distribution< _RealType >, 3270
parse_numbers.h, 3357
partial_sort
Sorting, 80
partial_sort_copy
Sorting, 81
partial_sort_minimal_n
__gnu_parallel::_Settings, 1034
partial_sum
Generalized Numeric operations, 14
partial_sum.h, 3667
partial_sum_dilation
__gnu_parallel::_Settings, 1034
partial_sum_minimal_n
__gnu_parallel::_Settings, 1034
partition
Mutating, 29
partition.h, 3668
_GLIBCXX_VOLATILE, 3668
partition_chunk_share
__gnu_parallel::_Settings, 1034
partition_chunk_size
__gnu_parallel::_Settings, 1034
partition_copy
Mutating, 30
partition_minimal_n
__gnu_parallel::_Settings, 1034
partition_point
Mutating, 30
pat_trie_.hpp, 3571
pat_trie_base.hpp, 3572
pbackfail
__gnu_cxx::enc_filebuf< _CharT >, 2119
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2946
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2964
std::basic_filebuf< _CharT, _Traits >, 1121
std::basic_streambuf< _CharT, _Traits >, 1493
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1700
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3257
pbase
__gnu_cxx::enc_filebuf< _CharT >, 2119
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2947
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2964
std::basic_filebuf< _CharT, _Traits >, 1121
std::basic_streambuf< _CharT, _Traits >, 1494
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1701
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3257
pbump
__gnu_cxx::enc_filebuf< _CharT >, 2119
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2947
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2965
std::basic_filebuf< _CharT, _Traits >, 1122
std::basic_streambuf< _CharT, _Traits >, 1494
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1701
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3257
peek
std::basic_fstream< _CharT, _Traits >, 1167
std::basic_ifstream< _CharT, _Traits >, 1212
std::basic_iostream< _CharT, _Traits >, 1281
std::basic_istream< _CharT, _Traits >, 1323
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1363
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1744
perm_options
Filesystem, 166
perms
Filesystem, 166
Filesystem TS, 285
philox4x32
Random Number Generators, 249
philox4x64
Random Number Generators, 249
philox_engine
std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2750

piecewise_construct
Utilities, 313
pod_char_traits.h, 3585
point_const_iterator.hpp, 3552, 3553
point_iterator.hpp, 3581
point_iterators.hpp, 3540
pointer
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1782
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1785
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2337
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_iterator_< Node, _Alloc >, 2341
std::allocator_traits< _Alloc >, 1056
std::allocator_traits< allocator< _Tp > >, 1061
std::allocator_traits< allocator< void > >, 1066
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1070
std::back_insert_iterator< _Container >, 1099
std::front_insert_iterator< _Container >, 2193
std::insert_iterator< _Container >, 2283
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2323
std::istreambuf_iterator< _CharT, _Traits >, 2326
std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, 2332
std::ostream_iterator< _Tp, _CharT, _Traits >, 2691
std::ostreambuf_iterator< _CharT, _Traits >, 2694
std::pointer_traits< _Ptr >, 2765
std::pointer_traits< _Tp * >, 2766, 2767
std::raw_storage_iterator< _OutputIterator, _Tp >, 2807
std::set< _Key, _Compare, _Alloc >, 2884
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3118
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3147
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3172
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3196
Pointer Abstractions, 334
allocate_shared, 337
atomic_compare_exchange_strong, 338
atomic_compare_exchange_strong_explicit, 339
atomic_compare_exchange_weak, 340
atomic_compare_exchange_weak_explicit, 341
atomic_exchange, 342
atomic_exchange_explicit, 342, 343
atomic_is_lock_free, 343, 344
atomic_load, 344
atomic_load_explicit, 345
atomic_store, 345, 346
atomic_store_explicit, 346, 347
const_pointer_cast, 347
dynamic_pointer_cast, 347
get_deleter, 348
make_shared, 348
operator<, 348, 349
operator<<, 349
operator<=, 349
operator>, 350
operator>=, 351
operator==, 349, 350
reinterpret_pointer_cast, 351
static_pointer_cast, 351
swap, 352
to_address, 352
pointer.h, 3586
pointer_to
std::pointer_traits< _Ptr >, 2765
std::pointer_traits< _Tp * >, 2767, 2768
Poisson Distributions, 241
operator<<, 242, 243
operator>>, 243, 244
polar
Complex Numbers, 189
std::tr1, 731
Policy-Based Data Structures, 143
policy_access_fn_imps.hpp, 3553, 3554
Polymorphic memory resources, 353
new_delete_resource, 354
pool_allocator.h, 3587
pop
std::priority_queue< _Tp, _Sequence, _Compare >, 2779
std::queue< _Tp, _Sequence >, 2788
std::stack< _Tp, _Sequence >, 2935
pop_back
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 867
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >, 977
std::basic_string< _CharT, _Traits, _Alloc >, 1661
std::deque< _Tp, _Alloc >, 2075
std::list< _Tp, _Alloc >, 2372
std::vector< _Tp, _Alloc >, 3242
pop_front
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >, 977
std::deque< _Tp, _Alloc >, 2075
std::forward_list< _Tp, _Alloc >, 2183
std::list< _Tp, _Alloc >, 2373
pop_heap
Heap, 93, 94
pos_format

std::moneypunct< _CharT, _Intl >, 2488
std::moneypunct_byname< _CharT, _Intl >, 2495
pos_type
std::basic_ios< _CharT, _Traits >, 1234
std::basic_streambuf< _CharT, _Traits >, 1490
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3253
position
std::match_results< _Bi_iter, _Alloc >, 2444
positive_sign
std::moneypunct< _CharT, _Intl >, 2488
std::moneypunct_byname< _CharT, _Intl >, 2496
postypes.h, 3357
pow
Complex Numbers, 189
math.h, 3626
power
SGI, 159
pptr
__gnu_cxx::enc_filebuf< _CharT >, 2120
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2947
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2965
std::basic_filebuf< _CharT, _Traits >, 1122
std::basic_streambuf< _CharT, _Traits >, 1494
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1701
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3258
precision
std::basic_fstream< _CharT, _Traits >, 1167
std::basic_ifstream< _CharT, _Traits >, 1212, 1213
std::basic_ios< _CharT, _Traits >, 1240, 1241
std::basic_iostream< _CharT, _Traits >, 1282
std::basic_istream< _CharT, _Traits >, 1323
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1363, 1364
std::basic_ofstream< _CharT, _Traits >, 1399
std::basic_ostream< _CharT, _Traits >, 1430, 1431
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1463, 1464
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1744
std::ios_base, 2293, 2294
predefined_ops.h, 3358
prefix
std::match_results< _Bi_iter, _Alloc >, 2444
prefix_range
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3065, 3066
prefix_search_node_update_imp.hpp, 3579
prev_permutation
Sorting, 82
print, 3675
print.h, 3358
priority_queue
__gnu_pbds::priority_queue< _Tv, Cmp_Fn, Tag, _Alloc >, 2774
std::priority_queue< _Tp, _Sequence, _Compare >, 2777, 2778
priority_queue.hpp, 3583
priority_queue_base_dispatch.hpp, 3573
probabilities
std::discrete_distribution< _IntType >, 2089
probe_fn_base.hpp, 3565
projected
std, 588
propagate_const, 3531
propagate_on_container_copy_assignment
__gnu_cxx::__alloc_traits< _Alloc, typename >, 750
std::allocator_traits< _Alloc >, 1056
std::allocator_traits< allocator< _Tp > >, 1061
std::allocator_traits< allocator< void > >, 1067
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1070
propagate_on_container_move_assignment
__gnu_cxx::__alloc_traits< _Alloc, typename >, 750
std::allocator_traits< _Alloc >, 1056
std::allocator_traits< allocator< _Tp > >, 1062
std::allocator_traits< allocator< void > >, 1067
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1070
propagate_on_container_swap
__gnu_cxx::__alloc_traits< _Alloc, typename >, 751
std::allocator_traits< _Alloc >, 1056
std::allocator_traits< allocator< _Tp > >, 1062
std::allocator_traits< allocator< void > >, 1067
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1071
ptr_fun
Adaptors for pointers to functions, 317
ptr_traits.h, 3358
pubimbue
__gnu_cxx::enc_filebuf< _CharT >, 2120
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2947
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2965
std::basic_filebuf< _CharT, _Traits >, 1122
std::basic_streambuf< _CharT, _Traits >, 1495
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1702
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3258
pubseekoff
__gnu_cxx::enc_filebuf< _CharT >, 2120
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2948
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2966
std::basic_filebuf< _CharT, _Traits >, 1123
std::basic_streambuf< _CharT, _Traits >, 1495
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1702
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3258

pubseekpos
__gnu_cxx::enc_filebuf< _CharT >, 2121
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2948
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2966
std::basic_filebuf< _CharT, _Traits >, 1123
std::basic_streambuf< _CharT, _Traits >, 1495
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1702
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3259
pubsetbuf
__gnu_cxx::enc_filebuf< _CharT >, 2121
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2948
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2966
std::basic_filebuf< _CharT, _Traits >, 1123
std::basic_streambuf< _CharT, _Traits >, 1496
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1703
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3259
pubsync
__gnu_cxx::enc_filebuf< _CharT >, 2121
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2949
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2966
std::basic_filebuf< _CharT, _Traits >, 1123
std::basic_streambuf< _CharT, _Traits >, 1496
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1703
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3259
push
std::priority_queue< _Tp, _Sequence, _Compare >, 2779
std::queue< _Tp, _Sequence >, 2788
std::stack< _Tp, _Sequence >, 2935
push_back
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 867
std::basic_string< _CharT, _Traits, _Alloc >, 1661, 1662
std::deque< _Tp, _Alloc >, 2075
std::list< _Tp, _Alloc >, 2373
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2107
std::vector< _Tp, _Alloc >, 3242
push_front
__gnu_parallel::_RestrictedBoundedConcurrentQueue< _Tp >, 978
std::deque< _Tp, _Alloc >, 2075
std::forward_list< _Tp, _Alloc >, 2183
std::list< _Tp, _Alloc >, 2373
push_heap
Heap, 94
put
std::basic_fstream< _CharT, _Traits >, 1167
std::basic_iostream< _CharT, _Traits >, 1282
std::basic_ofstream< _CharT, _Traits >, 1399
std::basic_ostream< _CharT, _Traits >, 1431
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1464
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1744
std::money_put< _CharT, _OutIter >, 2478, 2479
std::num_put< _CharT, _OutIter >, 2592–2597
std::time_put< _CharT, _OutIter >, 3035, 3036
std::time_put_byname< _CharT, _OutIter >, 3039
put_money
std, 636
put_time
std, 636
putback
std::basic_fstream< _CharT, _Traits >, 1168
std::basic_ifstream< _CharT, _Traits >, 1213
std::basic_iostream< _CharT, _Traits >, 1282
std::basic_istream< _CharT, _Traits >, 1323
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1364
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1745
pword
std::basic_fstream< _CharT, _Traits >, 1168
std::basic_ifstream< _CharT, _Traits >, 1213
std::basic_ios< _CharT, _Traits >, 1241
std::basic_iostream< _CharT, _Traits >, 1283
std::basic_istream< _CharT, _Traits >, 1324
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1364
std::basic_ofstream< _CharT, _Traits >, 1400
std::basic_ostream< _CharT, _Traits >, 1431
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1464
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1745
std::ios_base, 2294

qsb_steals
__gnu_parallel::_Settings, 1034
quadratic_probe_fn_imp.hpp, 3565
queue, 3675
std::queue< _Tp, _Sequence >, 2787
queue.h, 3668
_GLIBCXX_VOLATILE, 3669
quicksort.h, 3669
quiet_NaN
std::numeric_limits< _Tp >, 2600
quoted_string.h, 3359

r_erase_fn_imps.hpp, 3540
radix
std::__numeric_limits_base, 809
std::numeric_limits< _Tp >, 2602
std::numeric_limits< bool >, 2606
std::numeric_limits< char >, 2610

std::numeric_limits< char16_t >, 2614
std::numeric_limits< char32_t >, 2618
std::numeric_limits< double >, 2622
std::numeric_limits< float >, 2626
std::numeric_limits< int >, 2630
std::numeric_limits< long >, 2634
std::numeric_limits< long double >, 2638
std::numeric_limits< long long >, 2642
std::numeric_limits< short >, 2646
std::numeric_limits< signed char >, 2650
std::numeric_limits< unsigned char >, 2654
std::numeric_limits< unsigned int >, 2658
std::numeric_limits< unsigned long >, 2662
std::numeric_limits< unsigned long long >, 2666
std::numeric_limits< unsigned short >, 2670
std::numeric_limits< wchar_t >, 2674
random, 3675
Random Number Distributions, 237
Random Number Generation, 236
generate_canonical, 237
Random Number Generators, 247
minstd_rand, 248
minstd_rand0, 248
mt19937, 248
mt19937_64, 249
operator<<, 249
philox4x32, 249
philox4x64, 249
Random Number Utilities, 250
random.h, 3359
random.tcc, 3362, 3366
random_number.h, 3669
random_sample
SGI, 160
random_sample_n
SGI, 160
random_shuffle
Mutating, 31
random_shuffle.h, 3669
random_shuffle_minimal_n
__gnu_parallel::_Settings, 1034
range_access.h, 3368
ranged_hash_fn.hpp, 3566
ranged_probe_fn.hpp, 3566
Ranges, 255
ranges, 3676
ranges_algo.h, 3369
ranges_algobase.h, 3374
ranges_base.h, 3375
enable_view, 3378
ranges_cmp.h, 3378
ranges_uninitialized.h, 3378
ranges_util.h, 3378
ratio, 3679, 3681
ratio_add
Rational Arithmetic, 365
ratio_divide
Rational Arithmetic, 365
ratio_multiply
Rational Arithmetic, 365
ratio_subtract
Rational Arithmetic, 365
Rational Arithmetic, 363
ratio_add, 365
ratio_divide, 365
ratio_multiply, 365
ratio_subtract, 365
rb_tree, 3588
rb_tree_.hpp, 3574
rbegin
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 867
std, 637, 638
std::basic_string< _CharT, _Traits, _Alloc >, 1662
std::deque< _Tp, _Alloc >, 2076
std::list< _Tp, _Alloc >, 2373, 2374
std::map< _Key, _Tp, _Compare, _Alloc >, 2431
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2524
std::multiset< _Key, _Compare, _Alloc >, 2545
std::set< _Key, _Compare, _Alloc >, 2897
std::vector< _Tp, _Alloc >, 3242
rc.hpp, 3574
rc_binomial_heap_.hpp, 3574
rc_string_base.h, 3588
rdbuf
std::basic_fstream< _CharT, _Traits >, 1169
std::basic_ifstream< _CharT, _Traits >, 1214
std::basic_ios< _CharT, _Traits >, 1241, 1242
std::basic_iostream< _CharT, _Traits >, 1283, 1284
std::basic_istream< _CharT, _Traits >, 1324, 1325
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1365
std::basic_ofstream< _CharT, _Traits >, 1400, 1401
std::basic_ostream< _CharT, _Traits >, 1432
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1465
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1746
rdstate
std::basic_fstream< _CharT, _Traits >, 1169
std::basic_ifstream< _CharT, _Traits >, 1214
std::basic_ios< _CharT, _Traits >, 1242
std::basic_iostream< _CharT, _Traits >, 1284
std::basic_istream< _CharT, _Traits >, 1325
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1365
std::basic_ofstream< _CharT, _Traits >, 1401

std::basic_ostream< _CharT, _Traits >, 1433
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1465
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1746
read
std::basic_fstream< _CharT, _Traits >, 1170
std::basic_ifstream< _CharT, _Traits >, 1215
std::basic_iostream< _CharT, _Traits >, 1284
std::basic_istream< _CharT, _Traits >, 1325
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1366
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1747
readsome
std::basic_fstream< _CharT, _Traits >, 1170
std::basic_ifstream< _CharT, _Traits >, 1215
std::basic_iostream< _CharT, _Traits >, 1285
std::basic_istream< _CharT, _Traits >, 1326
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1366
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1747
ready
std::match_results< _Bi_iter, _Alloc >, 2444
rebind
std::pointer_traits< _Ptr >, 2765
std::pointer_traits< _Tp * >, 2767
reduce
Generalized Numeric operations, 15, 16
ref
std::reference_wrapper< _Tp >, 2825
reference
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1767
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1772
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1782
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1785
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2337
__gnu_pbds::detail::left_child_next_sibling_heap_node_pointer_< Node, _Alloc >, 2341
std::back_insert_iterator< _Container >, 1099
std::front_insert_iterator< _Container >, 2193
std::insert_iterator< _Container >, 2283
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2323
std::istreambuf_iterator< _CharT, _Traits >, 2326
std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, 2332
std::ostream_iterator< _Tp, _CharT, _Traits >, 2691
std::ostreambuf_iterator< _CharT, _Traits >, 2694
std::raw_storage_iterator< _OutputIterator, _Tp >, 2807
std::set< _Key, _Compare, _Alloc >, 2884
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3118
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3147
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3172
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3196
refwrap.h, 3379
regex, 3681
Regular Expressions, 259
regex.h, 3380
regex.tcc, 3382
regex_automaton.h, 3382
regex_automaton.tcc, 3383
regex_compiler.h, 3383
regex_compiler.tcc, 3384
regex_constants.h, 3384
regex_error
std::regex_error, 2826
regex_error.h, 3386
regex_executor.h, 3386
regex_executor.tcc, 3387
regex_iterator
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2828
regex_match
Regular Expressions, 263–267
regex_replace
Regular Expressions, 268–271
regex_scanner.h, 3387
regex_scanner.tcc, 3387
regex_search
Regular Expressions, 272–275
regex_token_iterator
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2831–2833
regex_traits
std::regex_traits< _Ch_type >, 2835
register_callback_
std::basic_fstream< _CharT, _Traits >, 1171
std::basic_ifstream< _CharT, _Traits >, 1216
std::basic_ios< _CharT, _Traits >, 1242
std::basic_iostream< _CharT, _Traits >, 1286
std::basic_istream< _CharT, _Traits >, 1327
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1367
std::basic_ofstream< _CharT, _Traits >, 1401
std::basic_ostream< _CharT, _Traits >, 1433

std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1466
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1748
std::ios_base, 2294
Regular Expressions, 256
cregex_token_iterator, 259
csub_match, 259
operator<<, 260
operator<=>, 260, 261
operator==, 261–263
regex, 259
regex_match, 263–267
regex_replace, 268–271
regex_search, 272–275
sregex_token_iterator, 259
ssub_match, 259
swap, 276
wcregex_token_iterator, 259
wcsub_match, 259
wregex, 259
wsregex_token_iterator, 259
wssub_match, 260
rehash
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3138
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3163
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3187
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3212
reinterpret_pointer_cast
Pointer Abstractions, 351
release
std::auto_ptr< _Tp >, 1096
std::unique_ptr< _Tp, _Dp >, 3103
std::unique_ptr< _Tp[], _Dp >, 3108
remove
Mutating, 32
std::forward_list< _Tp, _Alloc >, 2184
std::list< _Tp, _Alloc >, 2374
remove_all_extents_t
Metaprogramming, 360
remove_copy
Mutating, 32
remove_copy_if
Mutating, 33
remove_extent_t
Metaprogramming, 360
remove_if
Mutating, 33
std::forward_list< _Tp, _Alloc >, 2184
std::list< _Tp, _Alloc >, 2374
remove_pointer_t
Metaprogramming, 360
remove_reference_t
Metaprogramming, 360
rend
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 868
std, 638, 639
std::basic_string< _CharT, _Traits, _Alloc >, 1662, 1663
std::deque< _Tp, _Alloc >, 2076
std::list< _Tp, _Alloc >, 2374
std::map< _Key, _Tp, _Compare, _Alloc >, 2432
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2524
std::multiset< _Key, _Compare, _Alloc >, 2546
std::set< _Key, _Compare, _Alloc >, 2898
std::vector< _Tp, _Alloc >, 3243
replace
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 868–874
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1561–1571
Mutating, 34
std::basic_string< _CharT, _Traits, _Alloc >, 1663–1676
replace_copy
std, 639
replace_copy_if
Mutating, 34
replace_if
Mutating, 35
replace_minimal_n
__gnu_parallel::_Settings, 1034
requires_hosted.h, 3387
reserve
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 875
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1572, 1573
std::basic_string< _CharT, _Traits, _Alloc >, 1677, 1678
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3138
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3164
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3188
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3213
std::vector< _Tp, _Alloc >, 3243
reset
std::auto_ptr< _Tp >, 1096
std::bernoulli_distribution, 1763

std::binomial_distribution< _IntType >, 1793
std::bitset< _Nb >, 1811
std::cauchy_distribution< _RealType >, 1818
std::chi_squared_distribution< _RealType >, 1843
std::discrete_distribution< _IntType >, 2090
std::exponential_distribution< _RealType >, 2151
std::extreme_value_distribution< _RealType >, 2154
std::fisher_f_distribution< _RealType >, 2164
std::gamma_distribution< _RealType >, 2211
std::geometric_distribution< _IntType >, 2214
std::lognormal_distribution< _RealType >, 2400
std::negative_binomial_distribution< _IntType >, 2555
std::normal_distribution< _RealType >, 2562
std::piecewise_constant_distribution< _RealType >, 2755
std::piecewise_linear_distribution< _RealType >, 2758
std::poisson_distribution< _IntType >, 2771
std::student_t_distribution< _RealType >, 2981
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2107
std::uniform_int_distribution< _IntType >, 3094
std::uniform_real_distribution< _RealType >, 3097
std::unique_ptr< _Tp, _Dp >, 3103
std::unique_ptr< _Tp[], _Dp >, 3109
std::weibull_distribution< _RealType >, 3271
resetiosflags
std, 640
resize
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 875, 876
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1574
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2267
Numeric Arrays, 234
std::basic_string< _CharT, _Traits, _Alloc >, 1678, 1679
std::deque< _Tp, _Alloc >, 2076
std::forward_list< _Tp, _Alloc >, 2184, 2185
std::list< _Tp, _Alloc >, 2375
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2107
std::vector< _Tp, _Alloc >, 3243, 3244
resize_fn_imps.hpp, 3561
resize_no_store_hash_fn_imps.hpp, 3561
resize_policy.hpp, 3554
resize_store_hash_fn_imps.hpp, 3561
result_of_t
Metaprogramming, 360
result_type
__gnu_cxx::binary_compose< _Operation1, _Operation2, _Operation3 >, 1775
__gnu_cxx::equal_to< _Tp >, 2136
__gnu_cxx::project1st< _Arg1, _Arg2 >, 2781
__gnu_cxx::project2nd< _Arg1, _Arg2 >, 2781
__gnu_cxx::select1st< _Pair >, 2873
__gnu_cxx::select2nd< _Pair >, 2873
__gnu_cxx::subtractive_rng, 2992
__gnu_cxx::unary_compose< _Operation1, _Operation2 >, 3087
__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >, 897
__gnu_parallel::_EqualTo< _T1, _T2 >, 898
__gnu_parallel::_Less< _T1, _T2 >, 924
__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >, 926
__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >, 927
__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >, 953
__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >, 965
__gnu_parallel::__binder1st< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 758
__gnu_parallel::__binder2nd< _Operation, _FirstArgumentType, _SecondArgumentType, _ResultType >, 759
__gnu_parallel::__unary_negate< _Predicate, argument_type >, 826
std::bernoulli_distribution, 1761
std::binary_function< _Arg1, _Arg2, _Result >, 1778
std::binary_negate< _Predicate >, 1788
std::binder1st< _Operation >, 1789
std::binder2nd< _Operation >, 1791
std::binomial_distribution< _IntType >, 1792
std::cauchy_distribution< _RealType >, 1817
std::chi_squared_distribution< _RealType >, 1843
std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >, 1915
std::const_mem_fun1_t< _Ret, _Tp, _Arg >, 1916
std::const_mem_fun_ref_t< _Ret, _Tp >, 1917
std::const_mem_fun_t< _Ret, _Tp >, 1918
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2084
std::discrete_distribution< _IntType >, 2089
std::divides< _Tp >, 2092
std::divides< void >, 2093
std::equal_to< _Tp >, 2137
std::equal_to< void >, 2139
std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >, 2707
std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >, 2709
std::exponential_distribution< _RealType >, 2150
std::extreme_value_distribution< _RealType >, 2153
std::fisher_f_distribution< _RealType >, 2164
std::gamma_distribution< _RealType >, 2210

std::geometric_distribution< _IntType >, 2213
std::greater< _Tp >, 2227
std::greater< void >, 2229
std::greater_equal< _Tp >, 2230
std::greater_equal< void >, 2231
std::hash< __gnu_cxx::throw_value_limit >, 2239
std::hash< __gnu_cxx::throw_value_random >, 2240
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2272
std::less< _Tp >, 2344
std::less< void >, 2346
std::less_equal< _Tp >, 2347
std::less_equal< void >, 2348
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2351
std::logical_and< _Tp >, 2392
std::logical_and< void >, 2393
std::logical_not< _Tp >, 2394
std::logical_not< void >, 2395
std::logical_or< _Tp >, 2396
std::logical_or< void >, 2397
std::lognormal_distribution< _RealType >, 2399
std::mem_fun1_ref_t< _Ret, _Tp, _Arg >, 2449
std::mem_fun1_t< _Ret, _Tp, _Arg >, 2450
std::mem_fun_ref_t< _Ret, _Tp >, 2451
std::mem_fun_t< _Ret, _Tp >, 2452
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2456
std::minus< _Tp >, 2465
std::minus< void >, 2466
std::modulus< _Tp >, 2468
std::modulus< void >, 2469
std::multiplies< _Tp >, 2527
std::multiplies< void >, 2528
std::negate< _Tp >, 2552
std::negate< void >, 2553
std::negative_binomial_distribution< _IntType >, 2554
std::normal_distribution< _RealType >, 2561
std::not_equal_to< _Tp >, 2564
std::not_equal_to< void >, 2566
std::owner_less< shared_ptr< _Tp > >, 2708
std::owner_less< void >, 2709
std::owner_less< weak_ptr< _Tp > >, 2710
std::piecewise_constant_distribution< _RealType >, 2754
std::piecewise_linear_distribution< _RealType >, 2757
std::plus< _Tp >, 2760
std::pointer_to_binary_function< _Arg1, _Arg2, _Result >, 2763
std::pointer_to_unary_function< _Arg, _Result >, 2764
std::poisson_distribution< _IntType >, 2769
std::random_device, 2792
std::seed_seq, 2872
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2917
std::student_t_distribution< _RealType >, 2980
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2988
std::unary_function< _Arg, _Result >, 3088
std::unary_negate< _Predicate >, 3089
std::uniform_int_distribution< _IntType >, 3093
std::uniform_real_distribution< _RealType >, 3095
std::weibull_distribution< _RealType >, 3270
rethrow_exception
exception_ptr.h, 3322
Exceptions, 135
rethrow_if_nested
Exceptions, 136
rethrow_nested
std::nested_exception, 2557
return_temporary_buffer
std, 640
reverse
Mutating, 35
std::forward_list< _Tp, _Alloc >, 2185
std::list< _Tp, _Alloc >, 2375
reverse_copy
Mutating, 36
reverse_iteration
__gnu_pbds::container_traits< Cntnr >, 1933
reverse_iterator
std::reverse_iterator< _Iterator >, 2846
std::set< _Key, _Compare, _Alloc >, 2884
rfind
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 876–878
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1574–1578
std::basic_string< _CharT, _Traits, _Alloc >, 1679–1682
riemann_zeta
Mathematical Special Functions, 216
TR1 Mathematical Special Functions, 254
riemann_zetaf
Mathematical Special Functions, 217
riemann_zetal
Mathematical Special Functions, 217
right
std, 640
std::basic_fstream< _CharT, _Traits >, 1182
std::basic_ifstream< _CharT, _Traits >, 1225
std::basic_ios< _CharT, _Traits >, 1250

std::basic_iostream< _CharT, _Traits >, 1297
std::basic_istream< _CharT, _Traits >, 1336
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1377
std::basic_ofstream< _CharT, _Traits >, 1410
std::basic_ostream< _CharT, _Traits >, 1442
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1475
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1759
std::ios_base, 2300
rope, 3589
ropeimpl.h, 3592
rotate
Mutating, 36
rotate_copy
Mutating, 37
rotate_fn_imps.hpp, 3540
round
Time, 384, 385
round_error
std::numeric_limits< _Tp >, 2600
round_indeterminate
std, 589
round_style
std::__numeric_limits_base, 809
std::numeric_limits< _Tp >, 2603
std::numeric_limits< bool >, 2606
std::numeric_limits< char >, 2610
std::numeric_limits< char16_t >, 2614
std::numeric_limits< char32_t >, 2618
std::numeric_limits< double >, 2622
std::numeric_limits< float >, 2626
std::numeric_limits< int >, 2630
std::numeric_limits< long >, 2634
std::numeric_limits< long double >, 2638
std::numeric_limits< long long >, 2642
std::numeric_limits< short >, 2646
std::numeric_limits< signed char >, 2650
std::numeric_limits< unsigned char >, 2654
std::numeric_limits< unsigned int >, 2658
std::numeric_limits< unsigned long >, 2662
std::numeric_limits< unsigned long long >, 2666
std::numeric_limits< unsigned short >, 2670
std::numeric_limits< wchar_t >, 2674
round_to_nearest
std, 589
round_toward_infinity
std, 589
round_toward_neg_infinity
std, 589
round_toward_zero
std, 589
runtime_error
std::runtime_error, 2855

safe_base.h, 3514
safe_container.h, 3514
safe_iterator.h, 3514
safe_iterator.tcc, 3515
safe_local_iterator.h, 3516
safe_local_iterator.tcc, 3517
safe_sequence.h, 3517
safe_sequence.tcc, 3517
safe_unordered_base.h, 3518
safe_unordered_container.h, 3518
safe_unordered_container.tcc, 3518
sample
std, 640
std::experimental, 707
sample_probe_fn
__gnu_pbds::sample_probe_fn, 2856
sample_probe_fn.hpp, 3566
sample_range_hashing
__gnu_pbds::sample_range_hashing, 2857
__gnu_pbds::sample_resize_policy, 2861
__gnu_pbds::sample_resize_trigger, 2864
__gnu_pbds::sample_size_policy, 2865
sample_range_hashing.hpp, 3567
sample_ranged_hash_fn
__gnu_pbds::sample_ranged_hash_fn, 2858
sample_ranged_hash_fn.hpp, 3567
sample_ranged_probe_fn.hpp, 3567
sample_resize_policy
__gnu_pbds::sample_resize_policy, 2860
sample_resize_policy.hpp, 3575
sample_resize_trigger
__gnu_pbds::sample_resize_trigger, 2862
sample_resize_trigger.hpp, 3576
sample_size_policy
__gnu_pbds::sample_size_policy, 2865
sample_size_policy.hpp, 3576
sample_tree_node_update.hpp, 3579
sample_trie_access_traits.hpp, 3579
sample_trie_node_update
__gnu_pbds::sample_trie_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 2867
sample_trie_node_update.hpp, 3579
sample_update_policy
__gnu_pbds::sample_update_policy, 2868
sample_update_policy.hpp, 3570
Sampling Distributions, 244
sat_arith.h, 3388
saturating_add
std, 641
saturating_cast
std, 641
saturating_div

std, 641
saturating_mul
std, 641
saturating_sub
std, 641
sbumpc
__gnu_cxx::enc_filebuf< _CharT >, 2121
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2949
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2967
std::basic_filebuf< _CharT, _Traits >, 1124
std::basic_streambuf< _CharT, _Traits >, 1496
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1703
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3259
scan_is
std::__ctype_abstract_base< _CharT >, 776
std::ctype< _CharT >, 1953
std::ctype< char >, 1978
std::ctype< wchar_t >, 1998
std::ctype_byname< _CharT >, 2012
std::ctype_byname< char >, 2033
scan_not
std::__ctype_abstract_base< _CharT >, 777
std::ctype< _CharT >, 1954
std::ctype< char >, 1979, 1980
std::ctype< wchar_t >, 1998
std::ctype_byname< _CharT >, 2013
std::ctype_byname< char >, 2034
scientific
std, 641
std::basic_fstream< _CharT, _Traits >, 1182
std::basic_ifstream< _CharT, _Traits >, 1225
std::basic_ios< _CharT, _Traits >, 1250
std::basic_iostream< _CharT, _Traits >, 1297
std::basic_istream< _CharT, _Traits >, 1336
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1377
std::basic_ofstream< _CharT, _Traits >, 1410
std::basic_ostream< _CharT, _Traits >, 1442
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1475
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1759
std::ios_base, 2300
scoped_allocator, 3682
search
Non-Mutating, 59, 60
std, 641
std::__parallel, 678
search.h, 3670
search_minimal_n
__gnu_parallel::_Settings, 1035
search_n
Non-Mutating, 61
second
__gnu_cxx::pair< _T1, _T2 >, 2716
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 920
std::pair< _T1, _T2 >, 2722
second_argument_type
__gnu_cxx::equal_to< _Tp >, 2136
__gnu_cxx::project1st< _Arg1, _Arg2 >, 2781
__gnu_cxx::project2nd< _Arg1, _Arg2 >, 2782
__gnu_parallel::_EqualFromLess< _T1, _T2, _Compare >, 897
__gnu_parallel::_EqualTo< _T1, _T2 >, 898
__gnu_parallel::_Less< _T1, _T2 >, 924
__gnu_parallel::_Lexicographic< _T1, _T2, _Compare >, 926
__gnu_parallel::_LexicographicReverse< _T1, _T2, _Compare >, 927
__gnu_parallel::_Multiplies< _Tp1, _Tp2, _Result >, 953
__gnu_parallel::_Plus< _Tp1, _Tp2, _Result >, 965
std::binary_function< _Arg1, _Arg2, _Result >, 1778
std::binary_negate< _Predicate >, 1788
std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >, 1915
std::const_mem_fun1_t< _Ret, _Tp, _Arg >, 1916
std::divides< _Tp >, 2092
std::divides< void >, 2093
std::equal_to< _Tp >, 2137
std::equal_to< void >, 2139
std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >, 2707
std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >, 2710
std::greater< _Tp >, 2227
std::greater< void >, 2229
std::greater_equal< _Tp >, 2230
std::greater_equal< void >, 2231
std::less< _Tp >, 2344
std::less< void >, 2346
std::less_equal< _Tp >, 2347
std::less_equal< void >, 2348
std::logical_and< _Tp >, 2392
std::logical_and< void >, 2393
std::logical_or< _Tp >, 2396
std::logical_or< void >, 2397
std::mem_fun1_ref_t< _Ret, _Tp, _Arg >, 2449
std::mem_fun1_t< _Ret, _Tp, _Arg >, 2450
std::minus< _Tp >, 2465
std::minus< void >, 2466
std::modulus< _Tp >, 2468
std::modulus< void >, 2469
std::multiplies< _Tp >, 2527
std::multiplies< void >, 2528
std::not_equal_to< _Tp >, 2564

std::not_equal_to< void >, 2566
std::owner_less< shared_ptr< _Tp > >, 2708
std::owner_less< void >, 2709
std::owner_less< weak_ptr< _Tp > >, 2710
std::plus< _Tp >, 2761
std::pointer_to_binary_function< _Arg1, _Arg2, _Result >, 2763
second_type
__gnu_cxx::pair< _T1, _T2 >, 2713
__gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 917
std::pair< _T1, _T2 >, 2719
seconds
Time, 375
seed
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2086
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2274
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2352
std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2751
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2919
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2989
seed_seq
std::seed_seq, 2872
seekdir
std::basic_fstream< _CharT, _Traits >, 1141
std::basic_ifstream< _CharT, _Traits >, 1192
std::basic_ios< _CharT, _Traits >, 1234
std::basic_iostream< _CharT, _Traits >, 1258
std::basic_istream< _CharT, _Traits >, 1305
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1345
std::basic_ofstream< _CharT, _Traits >, 1384
std::basic_ostream< _CharT, _Traits >, 1418
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1450
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1720
std::ios_base, 2291
seekg
std::basic_fstream< _CharT, _Traits >, 1171, 1172
std::basic_ifstream< _CharT, _Traits >, 1216, 1217
std::basic_iostream< _CharT, _Traits >, 1286
std::basic_istream< _CharT, _Traits >, 1327
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1367, 1368
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1748, 1749
seekoff
__gnu_cxx::enc_filebuf< _CharT >, 2121
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2949
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2967
std::basic_filebuf< _CharT, _Traits >, 1124
std::basic_streambuf< _CharT, _Traits >, 1496
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1703
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3260
seekp
std::basic_fstream< _CharT, _Traits >, 1172, 1173
std::basic_iostream< _CharT, _Traits >, 1287
std::basic_ofstream< _CharT, _Traits >, 1402
std::basic_ostream< _CharT, _Traits >, 1433, 1434
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1466
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1749
seekpos
__gnu_cxx::enc_filebuf< _CharT >, 2122
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2949
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2967
std::basic_filebuf< _CharT, _Traits >, 1124
std::basic_streambuf< _CharT, _Traits >, 1497
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1704
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3260
select_on_container_copy_construction
__gnu_cxx::__alloc_traits< _Alloc, typename >, 755
std::allocator_traits< _Alloc >, 1059
std::allocator_traits< allocator< _Tp > >, 1065
std::allocator_traits< allocator< void > >, 1068
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1074
semaphore, 3682
semaphore_base.h, 3388
sentry
std::basic_istream< _CharT, _Traits >::sentry, 2875
std::basic_ostream< _CharT, _Traits >::sentry, 2876
Sequences, 127
sequential
__gnu_parallel, 431
set, 3682, 3683
__gnu_parallel::_Settings, 1031
std::bitset< _Nb >, 1811
std::set< _Key, _Compare, _Alloc >, 2885–2887
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2107
Set Operations, 96
includes, 97
set_difference, 98, 99
set_intersection, 99, 100
set_symmetric_difference, 101
set_union, 102, 103
set.h, 3518
set_counter

std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2751
set_default_resource
memory_resource, 3634
set_difference
Set Operations, 98, 99
set_difference_minimal_n
__gnu_parallel::_Settings, 1035
set_intersection
Set Operations, 99, 100
set_intersection_minimal_n
__gnu_parallel::_Settings, 1035
set_load
__gnu_pbds::cc_hash_max_collision_check_resize_trigger< External_Load_Access, Size_Type >, 1822
set_loads
__gnu_pbds::hash_load_check_resize_trigger< External_Load_Access, Size_Type >, 2256
__gnu_pbds::hash_standard_resize_policy< Size_Policy, Trigger_Policy, External_Size_Access, Size_Type >, 2267
set_new_handler
std, 642
set_num_threads
__gnu_parallel::balanced_quicksort_tag, 1108
__gnu_parallel::balanced_tag, 1109
__gnu_parallel::default_parallel_tag, 2050
__gnu_parallel::exact_tag, 2145
__gnu_parallel::multiway_mergesort_exact_tag, 2548
__gnu_parallel::multiway_mergesort_sampling_tag, 2549
__gnu_parallel::multiway_mergesort_tag, 2550
__gnu_parallel::omp_loop_static_tag, 2685
__gnu_parallel::omp_loop_tag, 2686
__gnu_parallel::parallel_tag, 2726
__gnu_parallel::quicksort_tag, 2789
__gnu_parallel::sampling_tag, 2870
__gnu_parallel::unbalanced_tag, 3090
set_operations.h, 3670
set_symmetric_difference
Set Operations, 101
set_symmetric_difference_minimal_n
__gnu_parallel::_Settings, 1035
set_terminate
Exceptions, 136
set_union
Set Operations, 102, 103
set_union_minimal_n
__gnu_parallel::_Settings, 1035
setbase
std, 642
setbuf
__gnu_cxx::enc_filebuf< _CharT >, 2122
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2950
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2967
std::basic_filebuf< _CharT, _Traits >, 1124
std::basic_streambuf< _CharT, _Traits >, 1497
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1704
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3260
setf
std::basic_fstream< _CharT, _Traits >, 1173
std::basic_ifstream< _CharT, _Traits >, 1217
std::basic_ios< _CharT, _Traits >, 1243
std::basic_iostream< _CharT, _Traits >, 1288
std::basic_istream< _CharT, _Traits >, 1328
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1368
std::basic_ofstream< _CharT, _Traits >, 1402, 1403
std::basic_ostream< _CharT, _Traits >, 1434
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1467
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1750
std::ios_base, 2295
setfill
std, 642
setg
__gnu_cxx::enc_filebuf< _CharT >, 2122
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2950
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2968
std::basic_filebuf< _CharT, _Traits >, 1125
std::basic_streambuf< _CharT, _Traits >, 1497
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1704
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3261
setiosflags
std, 642
setp
__gnu_cxx::enc_filebuf< _CharT >, 2123
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2951
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2968
std::basic_filebuf< _CharT, _Traits >, 1125
std::basic_streambuf< _CharT, _Traits >, 1498
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1705
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3261
setprecision
std, 643
setstate
std::basic_fstream< _CharT, _Traits >, 1174
std::basic_ifstream< _CharT, _Traits >, 1218
std::basic_ios< _CharT, _Traits >, 1243
std::basic_iostream< _CharT, _Traits >, 1288
std::basic_istream< _CharT, _Traits >, 1328
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1369

std::basic_ofstream< _CharT, _Traits >, 1403
std::basic_ostream< _CharT, _Traits >, 1435
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1467
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1750
settings.h, 3671
_GLIBCXX_PARALLEL_CONDITION, 3672
setw
std, 643
sgetc
__gnu_cxx::enc_filebuf< _CharT >, 2123
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2951
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2969
std::basic_filebuf< _CharT, _Traits >, 1126
std::basic_streambuf< _CharT, _Traits >, 1498
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1705
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3261
sgetn
__gnu_cxx::enc_filebuf< _CharT >, 2123
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2951
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2969
std::basic_filebuf< _CharT, _Traits >, 1126
std::basic_streambuf< _CharT, _Traits >, 1498
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1705
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3262
SGI, 152
_Find_first, 156
_Find_next, 156
_Unchecked_flip, 156
_Unchecked_reset, 157
_Unchecked_set, 157
_Unchecked_test, 157
__median, 155
compose1, 157
compose2, 157
constant0, 157
constant1, 158
constant2, 158
copy_n, 158
distance, 158
identity_element, 159
lexicographical_compare_3way, 159
power, 159
random_sample, 160
random_sample_n, 160
uninitialized_copy_n, 161
shared_future
std::shared_future< _Res >, 2900
std::shared_future< _Res & >, 2902
std::shared_future< void >, 2904
shared_mutex, 3683
shared_ptr
std::shared_ptr< _Tp >, 2908–2914
shared_ptr.h, 3388, 3389
shared_ptr_atomic.h, 3391
shared_ptr_base.h, 3393
shift
Numeric Arrays, 235
showbase
std, 643
std::basic_fstream< _CharT, _Traits >, 1182
std::basic_ifstream< _CharT, _Traits >, 1226
std::basic_ios< _CharT, _Traits >, 1250
std::basic_iostream< _CharT, _Traits >, 1297
std::basic_istream< _CharT, _Traits >, 1336
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1377
std::basic_ofstream< _CharT, _Traits >, 1410
std::basic_ostream< _CharT, _Traits >, 1442
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1475
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1759
std::ios_base, 2301
showmanyc
__gnu_cxx::enc_filebuf< _CharT >, 2124
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2952
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2969
std::basic_filebuf< _CharT, _Traits >, 1126
std::basic_streambuf< _CharT, _Traits >, 1499
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1706
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3262
showpoint
std, 643
std::basic_fstream< _CharT, _Traits >, 1182
std::basic_ifstream< _CharT, _Traits >, 1226
std::basic_ios< _CharT, _Traits >, 1250
std::basic_iostream< _CharT, _Traits >, 1297
std::basic_istream< _CharT, _Traits >, 1336
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1377
std::basic_ofstream< _CharT, _Traits >, 1410
std::basic_ostream< _CharT, _Traits >, 1442
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1475
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1760
std::ios_base, 2301
showpos
std, 643
std::basic_fstream< _CharT, _Traits >, 1183
std::basic_ifstream< _CharT, _Traits >, 1226
std::basic_ios< _CharT, _Traits >, 1250
std::basic_iostream< _CharT, _Traits >, 1297

std::basic_istream< _CharT, _Traits >, 1337
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1377
std::basic_ofstream< _CharT, _Traits >, 1411
std::basic_ostream< _CharT, _Traits >, 1442
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1475
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1760
std::ios_base, 2301
shrink_to_fit
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 878
std::basic_string< _CharT, _Traits, _Alloc >, 1683
std::deque< _Tp, _Alloc >, 2077
std::vector< _Tp, _Alloc >, 3244
shuffle
Mutating, 38
shuffle_order_engine
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2917, 2918
signaling_NaN
std::numeric_limits< _Tp >, 2600
simd, 3532
sin
Complex Numbers, 189
math.h, 3626
sinh
Complex Numbers, 190
math.h, 3626
size
__gnu_cxx::_Temporary_buffer< _ForwardIterator, _Tp >, 1040
__gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 878
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 2997
__gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1579
Numeric Arrays, 235
std, 643, 644
std::_Temporary_buffer< _ForwardIterator, _Tp >, 1042
std::basic_string< _CharT, _Traits, _Alloc >, 1683
std::bitset< _Nb >, 1812
std::deque< _Tp, _Alloc >, 2077
std::list< _Tp, _Alloc >, 2375
std::map< _Key, _Tp, _Compare, _Alloc >, 2432
std::match_results< _Bi_iter, _Alloc >, 2445
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2525
std::multiset< _Key, _Compare, _Alloc >, 2546
std::priority_queue< _Tp, _Sequence, _Compare >, 2779
std::queue< _Tp, _Sequence >, 2788
std::set< _Key, _Compare, _Alloc >, 2898
std::stack< _Tp, _Sequence >, 2935
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2108
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3138
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3164
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3188
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3213
std::vector< _Tp, _Alloc >, 3244
size_fn_imps.hpp, 3561
size_type
__gnu_pbds::hash_prime_size_policy, 2262
__gnu_pbds::sample_range_hashing, 2857
__gnu_pbds::sample_resize_policy, 2860
__gnu_pbds::sample_resize_trigger, 2862
__gnu_pbds::sample_size_policy, 2865
__gnu_pbds::trie_prefix_search_node_update< Node_CItr, Node_Itr, _ATraits, _Alloc >, 3065
std::allocator_traits< _Alloc >, 1056
std::allocator_traits< allocator< _Tp > >, 1062
std::allocator_traits< allocator< void > >, 1067
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1071
std::set< _Key, _Compare, _Alloc >, 2884
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3118
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3147
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3172
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3196
skipws
std, 644
std::basic_fstream< _CharT, _Traits >, 1183
std::basic_ifstream< _CharT, _Traits >, 1226
std::basic_ios< _CharT, _Traits >, 1250
std::basic_iostream< _CharT, _Traits >, 1297
std::basic_istream< _CharT, _Traits >, 1337
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1377
std::basic_ofstream< _CharT, _Traits >, 1411
std::basic_ostream< _CharT, _Traits >, 1442
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1476
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1760
std::ios_base, 2301
sleep_for
std::this_thread, 725

sleep_until
std::this_thread, 725
slice
Numeric Arrays, 223, 224
slice_array
Numeric Arrays, 224
slice_array.h, 3394
slist, 3598
snextc
__gnu_cxx::enc_filebuf< _CharT >, 2124
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2952
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2970
std::basic_filebuf< _CharT, _Traits >, 1127
std::basic_streambuf< _CharT, _Traits >, 1499
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1706
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3263
sort
Sorting, 83
std::forward_list< _Tp, _Alloc >, 2185
std::list< _Tp, _Alloc >, 2375, 2376
sort.h, 3672
sort_heap
Heap, 95
sort_minimal_n
__gnu_parallel::_Settings, 1035
sort_mwms_oversampling
__gnu_parallel::_Settings, 1035
sort_qs_num_samples_preset
__gnu_parallel::_Settings, 1035
sort_qsb_base_case_maximal_n
__gnu_parallel::_Settings, 1035
Sorting, 62
clamp, 65
inplace_merge, 66
is_sorted, 67
is_sorted_until, 68
lexicographical_compare, 69
lexicographical_compare_three_way, 70
max, 70, 71
max_element, 71, 72
merge, 72, 73
min, 74
min_element, 75
minmax, 76
minmax_element, 77
next_permutation, 78
nth_element, 79
partial_sort, 80
partial_sort_copy, 81
prev_permutation, 82
sort, 83
stable_sort, 84
source_location, 3683
span, 3683
specfun.h, 3394
sph_bessel
Mathematical Special Functions, 217
TR1 Mathematical Special Functions, 255
sph_besself
Mathematical Special Functions, 218
sph_bessell
Mathematical Special Functions, 218
sph_legendre
Mathematical Special Functions, 218
TR1 Mathematical Special Functions, 255
sph_legendref
Mathematical Special Functions, 219
sph_legendrel
Mathematical Special Functions, 219
sph_neumann
Mathematical Special Functions, 219
TR1 Mathematical Special Functions, 255
sph_neumannf
Mathematical Special Functions, 220
sph_neumannl
Mathematical Special Functions, 220
splay_fn_imps.hpp, 3576
splay_tree_.hpp, 3576
splice
std::list< _Tp, _Alloc >, 2376, 2377
splice_after
std::forward_list< _Tp, _Alloc >, 2185–2187
split_fn_imps.hpp, 3573
split_join_can_throw
__gnu_pbds::container_traits< Cntnr >, 1933
split_join_fn_imps.hpp, 3554, 3555
sputbackc
__gnu_cxx::enc_filebuf< _CharT >, 2124
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2952
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2970
std::basic_filebuf< _CharT, _Traits >, 1127
std::basic_streambuf< _CharT, _Traits >, 1499
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1706
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3263
sputc
__gnu_cxx::enc_filebuf< _CharT >, 2125
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2953
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2970
std::basic_filebuf< _CharT, _Traits >, 1127
std::basic_streambuf< _CharT, _Traits >, 1500
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1707
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3263
sputn
__gnu_cxx::enc_filebuf< _CharT >, 2125
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2953

__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2971
std::basic_filebuf< _CharT, _Traits >, 1128
std::basic_streambuf< _CharT, _Traits >, 1500
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1707
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3264
sqrt
Complex Numbers, 190
math.h, 3626
sregex_token_iterator
Regular Expressions, 259
ssize
std, 644, 645
sso_string_base.h, 3599
sstream, 3684
sstream.tcc, 3396
ssub_match
Regular Expressions, 259
stable_partition
Mutating, 38
stable_sort
Sorting, 84
stack, 3684
std::stack< _Tp, _Sequence >, 2935
standard_policies.hpp, 3577
start
Numeric Arrays, 235, 236
state
std::fpos< _StateT >, 2189, 2190
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3264
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3274
static_pointer_cast
Pointer Abstractions, 351
std, 645
std, 475
_Construct, 590
_Destroy, 590, 591
_Destroy_n, 591
__gen_two_uniform_ints, 590
__lg, 590
__ptr_rebind, 587
__umap_traits, 587
__ummap_traits, 587
__umset_traits, 587
__uset_traits, 587
abs, 591
acosh, 591
advance, 591
begin, 592
boolalpha, 593
cbegin, 593
cend, 593
cerr, 649
chars_format, 589
cin, 649
clog, 649
compare_three_way_result_t, 587
const_pointer_cast, 593
cout, 649
crbegin, 594
crend, 594
data, 594, 595
dec, 595
default_sentinel, 649
defaultfloat, 595
denorm_absent, 589
denorm_indeterminate, 589
denorm_present, 589
destroying_delete, 649
distance, 595
dynamic_pointer_cast, 596
empty, 596
end, 597
endl, 598
ends, 598
fabs, 598
fixed, 598
float_denorm_style, 589
float_round_style, 589
flush, 598
from_chars, 599
generate_canonical, 599
get, 599–601
get_money, 601
get_new_handler, 601
get_temporary_buffer, 601
get_time, 602
getline, 602–604
hex, 604
hexfloat, 604
ignore, 649
index_sequence, 587
index_sequence_for, 587
inout_ptr, 604
internal, 605
io_errc, 589
isalnum, 605
isalpha, 605
isblank, 605
iscntrl, 606
isdigit, 606
isgraph, 606
islower, 606
isprint, 606
ispunct, 606
isspace, 607
isupper, 607

isxdigit, 607
iter_rvalue_reference_t, 588
left, 607
make_index_sequence, 588
make_integer_sequence, 588
new_handler, 588
noboolalpha, 607
noshowbase, 607
noshowpoint, 607
noshowpos, 608
noskipws, 608
nounitbuf, 608
nouppercase, 608
oct, 608
operator!=, 608, 609
operator<, 611–613
operator<<, 613–619
operator<=, 619, 620
operator<=>, 620–623
operator>, 628, 629
operator>>, 630–635
operator>=, 629, 630
operator+, 609–611
operator==, 623–628
operator&, 609
operator^, 635
operator|, 635
out_ptr, 636
projected, 588
put_money, 636
put_time, 636
rbegin, 637, 638
rend, 638, 639
replace_copy, 639
resetiosflags, 640
return_temporary_buffer, 640
right, 640
round_indeterminate, 589
round_to_nearest, 589
round_toward_infinity, 589
round_toward_neg_infinity, 589
round_toward_zero, 589
sample, 640
saturating_add, 641
saturating_cast, 641
saturating_div, 641
saturating_mul, 641
saturating_sub, 641
scientific, 641
search, 641
set_new_handler, 642
setbase, 642
setfill, 642
setiosflags, 642
setprecision, 643
setw, 643
showbase, 643
showpoint, 643
showpos, 643
size, 643, 644
skipws, 644
ssize, 644, 645
static_pointer_cast, 645
streamoff, 588
streampos, 588
streamsize, 588
swap, 645–648
tolower, 648
toupper, 648
u16streampos, 588
u32streampos, 588
unitbuf, 648
uppercase, 648
wcerr, 650
wcin, 650
wclog, 650
wcout, 650
ws, 649
wstreampos, 589
std::__basic_future< _Res >, 755
_M_get_result, 757
std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 759
do_out, 760
in, 761
out, 762
unshift, 762
std::__ctype_abstract_base< _CharT >, 766
char_type, 768
do_is, 769
do_narrow, 769, 770
do_scan_is, 771
do_scan_not, 771
do_tolower, 772
do_toupper, 773
do_widen, 773, 774
is, 774, 775
narrow, 775, 776
scan_is, 776
scan_not, 777
tolower, 777, 778
toupper, 778
widen, 779
std::__cxx11::collate< _CharT >, 1894
~collate, 1896
char_type, 1895
collate, 1895, 1896
compare, 1896

do_compare, 1897
do_hash, 1897
do_transform, 1898
hash, 1898
id, 1899
string_type, 1895
transform, 1898
std::__cxx11::collate_byname< _CharT >, 1899
char_type, 1900
string_type, 1900
std::__debug, 650
swap, 657
std::__debug::bitset< _Nb >, 1801
std::__debug::deque< _Tp, _Allocator >, 2053
std::__debug::forward_list< _Tp, _Alloc >, 2167
std::__debug::list< _Tp, _Allocator >, 2355
std::__debug::map< _Key, _Tp, _Compare, _Allocator >, 2409
std::__debug::multimap< _Key, _Tp, _Compare, _Allocator >, 2504
std::__debug::multiset< _Key, _Compare, _Allocator >, 2528
std::__debug::set< _Key, _Compare, _Allocator >, 2878
std::__debug::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3110
std::__debug::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3139
std::__debug::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3165
std::__debug::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3188
std::__debug::vector< _Tp, _Allocator >, 3222
vector, 3225
std::__detail, 657
__from_chars_alnum, 660
__from_chars_pow2_base, 660
operator<<, 661
operator>>, 661
std::__detail::_BracketMatcher< _TraitsT, __icase, __collate >, 885
std::__detail::_Compiler< _TraitsT >, 888
std::__detail::_Executor< _BiIter, _Alloc, _TraitsT >, 898
std::__detail::_List_node_base, 932
std::__detail::_List_node_header, 933
std::__detail::_Quoted_string< _String, _CharT >, 972
std::__detail::_Scanner< _CharT >, 1028
_TokenT, 1029
std::__detail::_StateSeq< _TraitsT >, 1038
std::__fwdlist::_Iterator< _Const, _Ptr >, 913
operator==, 914
std::__fwdlist::_Node< _ValPtr >, 953
std::__fwdlist::_Node_base< _VoidPtr >, 955
std::__is_fast_hash< _Hash >, 795
std::__is_location_invariant< _Tp >, 796
std::__new_allocator< _Tp >, 804
std::__numeric_limits_base, 805
digits, 807
digits10, 807
has_denorm, 807
has_denorm_loss, 807
has_infinity, 808
has_quiet_NaN, 808
has_signaling_NaN, 808
is_bounded, 808
is_exact, 808
is_iec559, 808
is_integer, 808
is_modulo, 808
is_signed, 808
is_specialized, 808
max_digits10, 809
max_exponent, 809
max_exponent10, 809
min_exponent, 809
min_exponent10, 809
radix, 809
round_style, 809
tinyness_before, 809
traps, 809
std::__parallel, 661
search, 678
std::__parallel::_CRandNumber< _MustBeInt >, 889
std::__unspecified__::exception_ptr, 2147
std::_Base_bitset< 0 >, 881
_M_w, 882
std::_Base_bitset< 1 >, 882
std::_Base_bitset< _Nw >, 880
_M_w, 881
std::_Bind< _Signature >, 884
std::_Bind_result< _Result, _Signature >, 884
std::_Deque_base< _Tp, _Alloc >, 889
_M_initialize_map, 890
std::_Deque_iterator< _Tp, _Ref, _Ptr >, 891
_M_set_node, 892
std::_Function_base, 901
std::_Fwd_list_base< _Tp, _Alloc >, 901
std::_Fwd_list_const_iterator< _Tp >, 903
operator==, 904
std::_Fwd_list_iterator< _Tp >, 904
operator==, 905
std::_Fwd_list_node< _Tp >, 905
std::_Fwd_list_node_base, 906
std::_List_base< _Tp, _Alloc >, 927
std::_List_const_iterator< _Tp >, 929
std::_List_iterator< _Tp >, 929
std::_List_node< _Tp >, 930
std::_Not_fn< _Fn >, 962
std::_Placeholder< _Num >, 964

std::_Sp_ebo_helper< _Tp, false >, 1036
std::_Sp_ebo_helper< _Tp, true >, 1036
std::_Temporary_buffer< _ForwardIterator, _Tp >, 1040
_M_requested_size, 1042
_Temporary_buffer, 1041
begin, 1042
end, 1042
size, 1042
std::_Vector_base< _Tp, _Alloc >, 1043
std::add_const< _Tp >, 1044
std::add_cv< _Tp >, 1045
std::add_lvalue_reference< _Tp >, 1045
std::add_pointer< _Tp >, 1045
std::add_rvalue_reference< _Tp >, 1046
std::add_volatile< _Tp >, 1046
std::adopt_lock_t, 1046
std::aligned_storage< _Len, _Align >, 1047
std::aligned_union< _Len, _Types >, 1047
std::alignment_of< _Tp >, 1048
std::allocator< _Tp >, 1050
std::allocator< void >, 1051
operator==, 1052
std::allocator_traits< _Alloc >, 1053
allocate, 1057
allocate_at_least, 1058
allocator_type, 1055
const_pointer, 1055
const_void_pointer, 1055
construct, 1058
deallocate, 1058
destroy, 1059
difference_type, 1056
is_always_equal, 1056
max_size, 1059
pointer, 1056
propagate_on_container_copy_assignment, 1056
propagate_on_container_move_assignment, 1056
propagate_on_container_swap, 1056
select_on_container_copy_construction, 1059
size_type, 1056
value_type, 1057
void_pointer, 1057
std::allocator_traits< allocator< _Tp > >, 1060
allocate, 1062
allocate_at_least, 1063
allocator_type, 1061
const_pointer, 1061
const_void_pointer, 1061
construct, 1063
deallocate, 1064
destroy, 1064
difference_type, 1061
is_always_equal, 1061
max_size, 1064
pointer, 1061
propagate_on_container_copy_assignment, 1061
propagate_on_container_move_assignment, 1062
propagate_on_container_swap, 1062
select_on_container_copy_construction, 1065
size_type, 1062
value_type, 1062
void_pointer, 1062
std::allocator_traits< allocator< void > >, 1065
allocate, 1067
allocator_type, 1066
const_pointer, 1066
const_void_pointer, 1066
construct, 1067
deallocate, 1068
destroy, 1068
difference_type, 1066
is_always_equal, 1066
max_size, 1068
pointer, 1066
propagate_on_container_copy_assignment, 1067
propagate_on_container_move_assignment, 1067
propagate_on_container_swap, 1067
select_on_container_copy_construction, 1068
size_type, 1067
value_type, 1067
void_pointer, 1067
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1069
allocate, 1071
allocate_at_least, 1072
allocator_type, 1070
const_pointer, 1070
const_void_pointer, 1070
construct, 1072
deallocate, 1073
destroy, 1073
difference_type, 1070
is_always_equal, 1070
max_size, 1073
pointer, 1070
propagate_on_container_copy_assignment, 1070
propagate_on_container_move_assignment, 1070
propagate_on_container_swap, 1071
select_on_container_copy_construction, 1074
size_type, 1071
value_type, 1071
void_pointer, 1071
std::array< _Tp, _Nm >, 1078
std::assignable_from, 733
std::atomic< _Tp >, 1080
std::atomic< _Tp * >, 1081
std::atomic< bool >, 1082
std::atomic< char >, 1083

std::atomic< char16_t >, 1084
std::atomic< char32_t >, 1085
std::atomic< int >, 1085
std::atomic< long >, 1086
std::atomic< long long >, 1086
std::atomic< short >, 1087
std::atomic< signed char >, 1088
std::atomic< unsigned char >, 1088
std::atomic< unsigned int >, 1089
std::atomic< unsigned long >, 1089
std::atomic< unsigned long long >, 1090
std::atomic< unsigned short >, 1091
std::atomic< wchar_t >, 1091
std::atomic_flag, 1092
std::auto_ptr< _Tp >, 1092
~auto_ptr, 1094
auto_ptr, 1094, 1095
element_type, 1093
get, 1095
operator->, 1095
operator=, 1095, 1096
operator*, 1095
release, 1096
reset, 1096
std::auto_ptr_ref< _Tp1 >, 1097
std::back_insert_iterator< _Container >, 1097
back_insert_iterator, 1099
container_type, 1099
iterator_category, 1099
operator++, 1099
operator=, 1100
operator*, 1099
pointer, 1099
reference, 1099
value_type, 1099
std::bad_alloc, 1100
what, 1101
std::bad_cast, 1102
what, 1102
std::bad_exception, 1103
what, 1103
std::bad_function_call, 1103
what, 1104
std::bad_typeid, 1105
what, 1106
std::bad_weak_ptr, 1106
what, 1107
std::basic_filebuf< _CharT, _Traits >, 1111
_M_buf, 1131
_M_buf_locale, 1131
_M_buf_size, 1131
_M_create_pback, 1115
_M_destroy_pback, 1115
_M_ext_buf, 1131
_M_ext_buf_size, 1131
_M_ext_next, 1131
_M_in_beg, 1131
_M_in_cur, 1131
_M_in_end, 1132
_M_mode, 1132
_M_out_beg, 1132
_M_out_cur, 1132
_M_out_end, 1132
_M_pback, 1132
_M_pback_cur_save, 1132
_M_pback_end_save, 1133
_M_pback_init, 1133
_M_reading, 1133
_M_set_buffer, 1115
~basic_filebuf, 1115
basic_filebuf, 1115
close, 1116
eback, 1116
egptr, 1116
epptr, 1117
gbump, 1117
getloc, 1117
gptr, 1117
imbue, 1118
in_avail, 1118
is_open, 1118
open, 1119, 1120
overflow, 1120
pbackfail, 1121
pbase, 1121
pbump, 1122
pptr, 1122
pubimbue, 1122
pubseekoff, 1123
pubseekpos, 1123
pubsetbuf, 1123
pubsync, 1123
sbumpc, 1124
seekoff, 1124
seekpos, 1124
setbuf, 1124
setg, 1125
setp, 1125
sgetc, 1126
sgetn, 1126
showmanyc, 1126
snextc, 1127
sputbackc, 1127
sputc, 1127
sputn, 1128
sungetc, 1128
sync, 1128
uflow, 1129

underflow, 1129
xsgetn, 1129
xsputn, 1130
std::basic_fstream< _CharT, _Traits >, 1133
_M_gcount, 1178
_M_getloc, 1143
~basic_fstream, 1143
adjustfield, 1178
app, 1179
ate, 1179
bad, 1143
badbit, 1179
basefield, 1179
basic_fstream, 1142, 1143
beg, 1179
binary, 1179
boolalpha, 1179
clear, 1144
close, 1144
copyfmt, 1144
cur, 1180
dec, 1180
end, 1180
eof, 1144
eofbit, 1180
event, 1142
event_callback, 1140
exceptions, 1145
fail, 1145
failbit, 1180
fill, 1146
fixed, 1181
flags, 1146
floatfield, 1181
flush, 1147
fmtflags, 1140
gcount, 1147
get, 1147–1149
getline, 1150
getloc, 1151
good, 1151
goodbit, 1181
hex, 1181
ignore, 1152
imbue, 1153
in, 1181
init, 1153
internal, 1181
iostate, 1141
is_open, 1154
iword, 1154
left, 1182
narrow, 1154
oct, 1182
open, 1155
openmode, 1141
operator bool, 1156
operator!, 1156
operator<<, 1156–1161
operator>>, 1161–1166
out, 1182
peek, 1167
precision, 1167
put, 1167
putback, 1168
pword, 1168
rdbuf, 1169
rdstate, 1169
read, 1170
readsome, 1170
register_callback, 1171
right, 1182
scientific, 1182
seekdir, 1141
seekg, 1171, 1172
seekp, 1172, 1173
setf, 1173
setstate, 1174
showbase, 1182
showpoint, 1182
showpos, 1183
skipws, 1183
sync, 1174
sync_with_stdio, 1174
tellg, 1175
tellp, 1175
tie, 1175, 1176
trunc, 1183
unget, 1176
unitbuf, 1183
unsetf, 1176
uppercase, 1183
widen, 1177
width, 1177
write, 1178
xalloc, 1178
std::basic_ifstream< _CharT, _Traits >, 1186
_M_gcount, 1222
_M_getloc, 1194
__num_put_type, 1191
~basic_ifstream, 1194
adjustfield, 1222
app, 1222
ate, 1222
bad, 1194
badbit, 1222
basefield, 1222
basic_ifstream, 1193, 1194

beg, 1222
binary, 1223
boolalpha, 1223
clear, 1195
close, 1195
copyfmt, 1195
cur, 1223
dec, 1223
end, 1223
eof, 1195
eofbit, 1223
event, 1193
event_callback, 1191
exceptions, 1196
fail, 1196
failbit, 1223
fill, 1197
fixed, 1224
flags, 1197
floatfield, 1224
fmtflags, 1191
gcount, 1198
get, 1198–1200
getline, 1201
getloc, 1202
good, 1202
goodbit, 1224
hex, 1224
ignore, 1202, 1203
imbue, 1203
in, 1224
init, 1204
internal, 1225
iostate, 1192
is_open, 1204
iword, 1204
left, 1225
narrow, 1205
oct, 1225
open, 1205, 1206
openmode, 1192
operator bool, 1206
operator!, 1206
operator>>, 1207–1212
out, 1225
peek, 1212
precision, 1212, 1213
putback, 1213
pword, 1213
rdbuf, 1214
rdstate, 1214
read, 1215
readsome, 1215
register_callback, 1216
right, 1225
scientific, 1225
seekdir, 1192
seekg, 1216, 1217
setf, 1217
setstate, 1218
showbase, 1226
showpoint, 1226
showpos, 1226
skipws, 1226
sync, 1218
sync_with_stdio, 1219
tellg, 1219
tie, 1219
trunc, 1226
unget, 1220
unitbuf, 1226
unsetf, 1220
uppercase, 1226
widen, 1220
width, 1221
xalloc, 1221
std::basic_ios< _CharT, _Traits >, 1227
_M_getloc, 1235
__ctype_type, 1231
__num_get_type, 1231
__num_put_type, 1231
~basic_ios, 1234
adjustfield, 1246
app, 1246
ate, 1246
bad, 1235
badbit, 1247
basefield, 1247
basic_ios, 1234, 1235
beg, 1247
binary, 1247
boolalpha, 1247
char_type, 1232
clear, 1235
copyfmt, 1236
cur, 1247
dec, 1247
end, 1248
eof, 1236
eofbit, 1248
event, 1234
event_callback, 1232
exceptions, 1236
fail, 1237
failbit, 1248
fill, 1237
fixed, 1248
flags, 1238

floatfield, 1248
fmtflags, 1232
getloc, 1238
good, 1238
goodbit, 1248
hex, 1249
imbue, 1239
in, 1249
init, 1239
int_type, 1233
internal, 1249
iostate, 1233
iword, 1239
left, 1249
narrow, 1240
oct, 1249
off_type, 1233
openmode, 1233
operator bool, 1240
operator!, 1240
out, 1250
pos_type, 1234
precision, 1240, 1241
pword, 1241
rdbuf, 1241, 1242
rdstate, 1242
register_callback, 1242
right, 1250
scientific, 1250
seekdir, 1234
setf, 1243
setstate, 1243
showbase, 1250
showpoint, 1250
showpos, 1250
skipws, 1250
sync_with_stdio, 1244
tie, 1244
traits_type, 1234
trunc, 1251
unitbuf, 1251
unsetf, 1245
uppercase, 1251
widen, 1245
width, 1245, 1246
xalloc, 1246
std::basic_iostream< _CharT, _Traits >, 1251
_M_gcount, 1293
_M_getloc, 1259
~basic_iostream, 1259
adjustfield, 1293
app, 1293
ate, 1293
bad, 1259
badbit, 1294
basefield, 1294
basic_iostream, 1259
beg, 1294
binary, 1294
boolalpha, 1294
clear, 1260
copyfmt, 1260
cur, 1294
dec, 1294
end, 1295
eof, 1260
eofbit, 1295
event, 1259
event_callback, 1257
exceptions, 1260, 1261
fail, 1261
failbit, 1295
fill, 1261, 1262
fixed, 1295
flags, 1262
floatfield, 1295
flush, 1263
fmtflags, 1257
gcount, 1263
get, 1263–1265
getline, 1266
getloc, 1267
good, 1267
goodbit, 1295
hex, 1296
ignore, 1267, 1268
imbue, 1268
in, 1296
init, 1269
internal, 1296
iostate, 1258
iword, 1269
left, 1296
narrow, 1270
oct, 1296
openmode, 1258
operator bool, 1270
operator!, 1270
operator<<, 1270–1275
operator>>, 1276–1281
out, 1297
peek, 1281
precision, 1282
put, 1282
putback, 1282
pword, 1283
rdbuf, 1283, 1284
rdstate, 1284

read, 1284
readsome, 1285
register_callback, 1286
right, 1297
scientific, 1297
seekdir, 1258
seekg, 1286
seekp, 1287
setf, 1288
setstate, 1288
showbase, 1297
showpoint, 1297
showpos, 1297
skipws, 1297
sync, 1289
sync_with_stdio, 1289
tellg, 1289
tellp, 1290
tie, 1290
trunc, 1298
unget, 1291
unitbuf, 1298
unsetf, 1291
uppercase, 1298
widen, 1291
width, 1292
write, 1292
xalloc, 1293
std::basic_istream< _CharT, _Traits >, 1298
_M_gcount, 1332
_M_getloc, 1306
__num_put_type, 1303
~basic_istream, 1306
adjustfield, 1332
app, 1333
ate, 1333
bad, 1306
badbit, 1333
basefield, 1333
basic_istream, 1306
beg, 1333
binary, 1333
boolalpha, 1333
clear, 1306
copyfmt, 1307
cur, 1334
dec, 1334
end, 1334
eof, 1307
eofbit, 1334
event, 1305
event_callback, 1303
exceptions, 1307
fail, 1308
failbit, 1334
fill, 1308
fixed, 1335
flags, 1309
floatfield, 1335
fmtflags, 1304
gcount, 1309
get, 1310–1312
getline, 1312, 1313
getloc, 1313
good, 1314
goodbit, 1335
hex, 1335
ignore, 1314, 1315
imbue, 1315
in, 1335
init, 1316
internal, 1335
iostate, 1304
iword, 1316
left, 1336
narrow, 1316
oct, 1336
openmode, 1305
operator bool, 1317
operator!, 1317
operator>>, 1317–1322
out, 1336
peek, 1323
precision, 1323
putback, 1323
pword, 1324
rdbuf, 1324, 1325
rdstate, 1325
read, 1325
readsome, 1326
register_callback, 1327
right, 1336
scientific, 1336
seekdir, 1305
seekg, 1327
setf, 1328
setstate, 1328
showbase, 1336
showpoint, 1336
showpos, 1337
skipws, 1337
sync, 1329
sync_with_stdio, 1329
tellg, 1330
tie, 1330
trunc, 1337
unget, 1330
unitbuf, 1337

unsetf, 1331
uppercase, 1337
widen, 1331
width, 1332
xalloc, 1332
std::basic_istream< _CharT, _Traits >::sentry, 2874
operator bool, 2875
sentry, 2875
traits_type, 2875
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1337
_M_gcount, 1373
_M_getloc, 1346
__num_put_type, 1343
~basic_istringstream, 1346
adjustfield, 1373
app, 1373
ate, 1373
bad, 1347
badbit, 1374
basefield, 1374
basic_istringstream, 1346
beg, 1374
binary, 1374
boolalpha, 1374
clear, 1347
copyfmt, 1347
cur, 1374
dec, 1374
end, 1375
eof, 1348
eofbit, 1375
event, 1345
event_callback, 1343
exceptions, 1348
fail, 1349
failbit, 1375
fill, 1349
fixed, 1375
flags, 1349, 1350
floatfield, 1375
fmtflags, 1344
gcount, 1350
get, 1350–1352
getline, 1353
getloc, 1354
good, 1354
goodbit, 1375
hex, 1376
ignore, 1354, 1355
imbue, 1356
in, 1376
init, 1356
internal, 1376
iostate, 1344
iword, 1356
left, 1376
narrow, 1357
oct, 1376
openmode, 1345
operator bool, 1357
operator!, 1357
operator>>, 1358–1363
out, 1377
peek, 1363
precision, 1363, 1364
putback, 1364
pword, 1364
rdbuf, 1365
rdstate, 1365
read, 1366
readsome, 1366
register_callback, 1367
right, 1377
scientific, 1377
seekdir, 1345
seekg, 1367, 1368
setf, 1368
setstate, 1369
showbase, 1377
showpoint, 1377
showpos, 1377
skipws, 1377
str, 1369
sync, 1370
sync_with_stdio, 1370
tellg, 1370
tie, 1371
trunc, 1378
unget, 1371
unitbuf, 1378
unsetf, 1372
uppercase, 1378
widen, 1372
width, 1372, 1373
xalloc, 1373
std::basic_ofstream< _CharT, _Traits >, 1378
_M_getloc, 1386
__num_get_type, 1383
~basic_ofstream, 1386
adjustfield, 1407
app, 1407
ate, 1407
bad, 1386
badbit, 1407
basefield, 1407
basic_ofstream, 1385, 1386
beg, 1407
binary, 1407

boolalpha, 1407
clear, 1387
close, 1387
copyfmt, 1387
cur, 1408
dec, 1408
end, 1408
eof, 1387
eofbit, 1408
event, 1385
event_callback, 1383
exceptions, 1388
fail, 1388
failbit, 1408
fill, 1389
fixed, 1409
flags, 1389
floatfield, 1409
flush, 1390
fmtflags, 1383
getloc, 1390
good, 1390
goodbit, 1409
hex, 1409
imbue, 1390
in, 1409
init, 1391
internal, 1409
iostate, 1384
is_open, 1391
iword, 1391
left, 1410
narrow, 1392
oct, 1410
open, 1392, 1393
openmode, 1384
operator bool, 1393
operator!, 1393
operator<<, 1394–1399
out, 1410
precision, 1399
put, 1399
pword, 1400
rdbuf, 1400, 1401
rdstate, 1401
register_callback, 1401
right, 1410
scientific, 1410
seekdir, 1384
seekp, 1402
setf, 1402, 1403
setstate, 1403
showbase, 1410
showpoint, 1410
showpos, 1411
skipws, 1411
sync_with_stdio, 1403
tellp, 1404
tie, 1404
trunc, 1411
unitbuf, 1411
unsetf, 1405
uppercase, 1411
widen, 1405
width, 1405
write, 1406
xalloc, 1406
std::basic_ostream< _CharT, _Traits >, 1411
_M_getloc, 1419
__num_get_type, 1416
~basic_ostream, 1419
adjustfield, 1438
app, 1438
ate, 1438
bad, 1419
badbit, 1439
basefield, 1439
basic_ostream, 1419
beg, 1439
binary, 1439
boolalpha, 1439
clear, 1419
copyfmt, 1420
cur, 1439
dec, 1439
end, 1440
eof, 1420
eofbit, 1440
event, 1418
event_callback, 1416
exceptions, 1420
fail, 1421
failbit, 1440
fill, 1421
fixed, 1440
flags, 1422
floatfield, 1440
flush, 1422
fmtflags, 1417
getloc, 1423
good, 1423
goodbit, 1440
hex, 1441
imbue, 1423
in, 1441
init, 1423
internal, 1441
iostate, 1417

iword, 1424
left, 1441
narrow, 1424
oct, 1441
openmode, 1418
operator bool, 1425
operator!, 1425
operator<<, 1425–1430
out, 1442
precision, 1430, 1431
put, 1431
pword, 1431
rdbuf, 1432
rdstate, 1433
register_callback, 1433
right, 1442
scientific, 1442
seekdir, 1418
seekp, 1433, 1434
setf, 1434
setstate, 1435
showbase, 1442
showpoint, 1442
showpos, 1442
skipws, 1442
sync_with_stdio, 1435
tellp, 1435
tie, 1436
trunc, 1443
unitbuf, 1443
unsetf, 1436
uppercase, 1443
widen, 1436
width, 1437
write, 1437
xalloc, 1438
std::basic_ostream< _CharT, _Traits >::sentry, 2876
~sentry, 2876
operator bool, 2877
sentry, 2876
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1443
_M_getloc, 1452
__num_get_type, 1449
~basic_ostringstream, 1452
adjustfield, 1471
app, 1471
ate, 1472
bad, 1452
badbit, 1472
basefield, 1472
basic_ostringstream, 1451
beg, 1472
binary, 1472
boolalpha, 1472
clear, 1452
copyfmt, 1453
cur, 1472
dec, 1473
end, 1473
eof, 1453
eofbit, 1473
event, 1451
event_callback, 1449
exceptions, 1453
fail, 1454
failbit, 1473
fill, 1454
fixed, 1473
flags, 1455
floatfield, 1473
flush, 1455
fmtflags, 1449
getloc, 1456
good, 1456
goodbit, 1474
hex, 1474
imbue, 1456
in, 1474
init, 1456
internal, 1474
iostate, 1450
iword, 1457
left, 1474
narrow, 1457
oct, 1475
openmode, 1450
operator bool, 1458
operator!, 1458
operator<<, 1458–1463
out, 1475
precision, 1463, 1464
put, 1464
pword, 1464
rdbuf, 1465
rdstate, 1465
register_callback, 1466
right, 1475
scientific, 1475
seekdir, 1450
seekp, 1466
setf, 1467
setstate, 1467
showbase, 1475
showpoint, 1475
showpos, 1475
skipws, 1476
str, 1468
sync_with_stdio, 1468

tellp, 1469
tie, 1469
trunc, 1476
unitbuf, 1476
unsetf, 1469
uppercase, 1476
widen, 1470
width, 1470
write, 1471
xalloc, 1471
std::basic_regex< _Ch_type, _Rx_traits >, 1476
~basic_regex, 1481
assign, 1481–1483
basic_regex, 1478–1480
flags, 1484
getloc, 1484
imbue, 1484
mark_count, 1484
operator=, 1484, 1485
swap, 1486
std::basic_streambuf< _CharT, _Traits >, 1486
_M_buf_locale, 1503
_M_in_beg, 1503
_M_in_cur, 1503
_M_in_end, 1503
_M_out_beg, 1503
_M_out_cur, 1504
_M_out_end, 1504
__streambuf_type, 1489
~basic_streambuf, 1490
basic_streambuf, 1490
char_type, 1489
eback, 1490
egptr, 1491
epptr, 1491
gbump, 1491
getloc, 1492
gptr, 1492
imbue, 1492
in_avail, 1493
int_type, 1489
off_type, 1489
overflow, 1493
pbackfail, 1493
pbase, 1494
pbump, 1494
pos_type, 1490
pptr, 1494
pubimbue, 1495
pubseekoff, 1495
pubseekpos, 1495
pubsetbuf, 1496
pubsync, 1496
sbumpc, 1496
seekoff, 1496
seekpos, 1497
setbuf, 1497
setg, 1497
setp, 1498
sgetc, 1498
sgetn, 1498
showmanyc, 1499
snextc, 1499
sputbackc, 1499
sputc, 1500
sputn, 1500
sungetc, 1501
sync, 1501
traits_type, 1490
uflow, 1501
underflow, 1501
xsgetn, 1502
xsputn, 1502
std::basic_string< _CharT, _Traits, _Alloc >, 1581
__resize_and_overwrite, 1598
~basic_string, 1594, 1598
append, 1598–1603
assign, 1604–1610
at, 1610–1612
back, 1612
basic_string, 1591–1597
begin, 1613
c_str, 1613
capacity, 1613, 1614
cbegin, 1614
cend, 1614
clear, 1614
compare, 1614–1620
copy, 1620, 1621
crbegin, 1621, 1622
crend, 1622
data, 1622, 1623
empty, 1623
end, 1623
erase, 1624–1626
find, 1626–1629
find_first_not_of, 1630–1633
find_first_of, 1633–1636
find_last_not_of, 1636–1639
find_last_of, 1640–1643
front, 1643
get_allocator, 1644
insert, 1644–1653
length, 1654
max_size, 1654
npos, 1686
operator+=, 1654–1657
operator=, 1657–1659

operator[], 1660, 1661
pop_back, 1661
push_back, 1661, 1662
rbegin, 1662
rend, 1662, 1663
replace, 1663–1676
reserve, 1677, 1678
resize, 1678, 1679
rfind, 1679–1682
shrink_to_fit, 1683
size, 1683
substr, 1684
subview, 1685
swap, 1686
std::basic_string_view< _CharT, _Traits >, 1687
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1693
_M_buf_locale, 1711
_M_in_beg, 1711
_M_in_cur, 1711
_M_in_end, 1711
_M_mode, 1711
_M_out_beg, 1711
_M_out_cur, 1711
_M_out_end, 1711
basic_stringbuf, 1696, 1697
eback, 1697
egptr, 1697
epptr, 1698
gbump, 1698
getloc, 1698
gptr, 1699
imbue, 1699
in_avail, 1699
overflow, 1700
pbackfail, 1700
pbase, 1701
pbump, 1701
pptr, 1701
pubimbue, 1702
pubseekoff, 1702
pubseekpos, 1702
pubsetbuf, 1703
pubsync, 1703
sbumpc, 1703
seekoff, 1703
seekpos, 1704
setbuf, 1704
setg, 1704
setp, 1705
sgetc, 1705
sgetn, 1705
showmanyc, 1706
snextc, 1706
sputbackc, 1706
sputc, 1707
sputn, 1707
str, 1708
sungetc, 1708
sync, 1708
uflow, 1709
underflow, 1709
xsgetn, 1709
xsputn, 1710
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1712
_M_gcount, 1756
_M_getloc, 1721
~basic_stringstream, 1721
adjustfield, 1756
app, 1756
ate, 1756
bad, 1722
badbit, 1756
basefield, 1756
basic_stringstream, 1721
beg, 1756
binary, 1756
boolalpha, 1757
clear, 1722
copyfmt, 1722
cur, 1757
dec, 1757
end, 1757
eof, 1723
eofbit, 1757
event, 1720
event_callback, 1719
exceptions, 1723
fail, 1724
failbit, 1757
fill, 1724
fixed, 1758
flags, 1724, 1725
floatfield, 1758
flush, 1725
fmtflags, 1719
gcount, 1725
get, 1725–1727
getline, 1728
getloc, 1729
good, 1729
goodbit, 1758
hex, 1758
ignore, 1730
imbue, 1731
in, 1758
init, 1731
internal, 1759
iostate, 1720

iword, 1732
left, 1759
narrow, 1732
oct, 1759
openmode, 1720
operator bool, 1732
operator!, 1733
operator<<, 1733–1738
operator>>, 1738–1743
out, 1759
peek, 1744
precision, 1744
put, 1744
putback, 1745
pword, 1745
rdbuf, 1746
rdstate, 1746
read, 1747
readsome, 1747
register_callback, 1748
right, 1759
scientific, 1759
seekdir, 1720
seekg, 1748, 1749
seekp, 1749
setf, 1750
setstate, 1750
showbase, 1759
showpoint, 1760
showpos, 1760
skipws, 1760
str, 1751
sync, 1751
sync_with_stdio, 1752
tellg, 1752
tellp, 1752
tie, 1753
trunc, 1760
unget, 1753
unitbuf, 1760
unsetf, 1753
uppercase, 1760
widen, 1754
width, 1754
write, 1755
xalloc, 1755
std::bernoulli_distribution, 1760
bernoulli_distribution, 1761
max, 1762
min, 1762
operator(), 1762
operator==, 1763
p, 1762
param, 1762
reset, 1763
result_type, 1761
std::bernoulli_distribution::param_type, 2726
std::bidirectional_iterator_tag, 1763
std::binary_function< _Arg1, _Arg2, _Result >, 1776
first_argument_type, 1778
result_type, 1778
second_argument_type, 1778
std::binary_negate< _Predicate >, 1787
first_argument_type, 1788
result_type, 1788
second_argument_type, 1788
std::binder1st< _Operation >, 1788
argument_type, 1789
result_type, 1789
std::binder2nd< _Operation >, 1790
argument_type, 1791
result_type, 1791
std::binomial_distribution< _IntType >, 1791
max, 1792
min, 1792
operator<<, 1793
operator>>, 1794
operator(), 1792
operator==, 1794
p, 1793
param, 1793
reset, 1793
result_type, 1792
t, 1793
std::binomial_distribution< _IntType >::param_type, 2727
std::bitset< _Nb >, 1802
all, 1807
any, 1807
bitset, 1805, 1806
count, 1807
flip, 1807
none, 1808
operator<<, 1808
operator<<=, 1808
operator>>, 1809
operator>>=, 1809
operator==, 1809
operator&=, 1808
operator[], 1809
operator~, 1810
operator^=, 1810
operator|=, 1810
reset, 1811
set, 1811
size, 1812
test, 1812
to_string, 1812
to_ulong, 1812

std::bitset< _Nb >::reference, 2823
std::cauchy_distribution< _RealType >, 1816
max, 1817
min, 1817
operator(), 1817
operator==, 1818
param, 1818
reset, 1818
result_type, 1817
std::cauchy_distribution< _RealType >::param_type, 2727
std::char_traits< __gnu_cxx::character< _Value, _Int, _St > >, 1835
std::char_traits< _CharT >, 1834
std::char_traits< char >, 1837
std::char_traits< wchar_t >, 1839
std::chi_squared_distribution< _RealType >, 1841
max, 1843
min, 1843
operator<<, 1844
operator>>, 1844
operator(), 1843
operator==, 1844
param, 1843
reset, 1843
result_type, 1843
std::chi_squared_distribution< _RealType >::param_type, 2728
std::chrono, 679
std::chrono::duration< _Rep, _Period >, 2094
std::chrono::duration_values< _Rep >, 2096
std::chrono::gps_clock, 2226
std::chrono::hh_mm_ss< _Duration >, 2268
std::chrono::steady_clock, 2974
std::chrono::system_clock, 2993
std::chrono::tai_clock, 2994
std::chrono::time_point< _Clock, _Dur >, 3032
std::chrono::treat_as_floating_point< _Rep >, 3041
std::chrono::tzdb_list, 3085
erase_after, 3086
front, 3086
std::chrono::tzdb_list::const_iterator, 1914
std::chrono::utc_clock, 3215
std::codecvt< _InternT, _ExternT, _StateT >, 1845
do_always_noconv, 1846
do_encoding, 1846
do_in, 1847
do_length, 1847
do_max_length, 1847
do_out, 1847
do_unshift, 1847
in, 1848
out, 1848
unshift, 1849
std::codecvt< _InternT, _ExternT, encoding_state >, 1850
do_always_noconv, 1852
do_encoding, 1852
do_in, 1852
do_length, 1852
do_max_length, 1852
do_out, 1853
do_unshift, 1853
in, 1853, 1854
out, 1855
unshift, 1856, 1857
std::codecvt< char, char, mbstate_t >, 1858
do_always_noconv, 1859
do_encoding, 1859
do_in, 1860
do_length, 1860
do_max_length, 1860
do_out, 1860
do_unshift, 1860
in, 1861
out, 1862, 1863
unshift, 1864
std::codecvt< char16_t, char, mbstate_t >, 1865
do_always_noconv, 1867
do_encoding, 1867
do_in, 1867
do_length, 1867
do_max_length, 1867
do_out, 1867
do_unshift, 1868
in, 1868, 1869
out, 1870
unshift, 1871, 1872
std::codecvt< char32_t, char, mbstate_t >, 1872
do_always_noconv, 1874
do_encoding, 1874
do_in, 1874
do_length, 1875
do_max_length, 1875
do_out, 1875
do_unshift, 1875
in, 1875, 1876
out, 1877, 1878
unshift, 1879
std::codecvt< wchar_t, char, mbstate_t >, 1880
do_always_noconv, 1882
do_encoding, 1882
do_in, 1882
do_length, 1882
do_max_length, 1882
do_out, 1882
do_unshift, 1883
in, 1883, 1884
out, 1885

unshift, 1886, 1887
std::codecvt_base, 1887
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1888
do_always_noconv, 1890
do_encoding, 1890
do_in, 1890
do_length, 1891
do_max_length, 1891
do_out, 1891
do_unshift, 1891
in, 1892
out, 1892
unshift, 1893
std::common_iterator< _It, _Sent >, 1900
std::common_reference_with, 733
std::common_type< _Tp >, 1901
std::common_type< chrono::duration< _Rep, _Period > >, 1901
std::common_type< chrono::duration< _Rep, _Period >, chrono::duration< _Rep, _Period > >, 1902
std::common_type< chrono::duration< _Rep1, _Period1 >, chrono::duration< _Rep2, _Period2 > >, 1902
std::common_type< chrono::time_point< _Clock, _Duration > >, 1902
std::common_type< chrono::time_point< _Clock, _Duration >, chrono::time_point< _Clock, _Duration > >, 1902
std::common_type< chrono::time_point< _Clock, _Duration1 >, chrono::time_point< _Clock, _Duration2 > >, 1903
std::common_with, 734
std::compare_three_way_result< _Tp, _Up >, 1903
std::complex< _Tp >, 1903
complex, 1904
explicit, 1905
operator+=, 1905
operator-=, 1905
operator/=, 1906
operator=, 1906
operator*=, 1905
value_type, 1904
std::complex< double >, 1906
std::complex< float >, 1907
std::complex< long double >, 1908
std::condition_variable, 1911
std::condition_variable_any, 1911
std::conditional< _Cond, _Iftrue, _Iffalse >, 1912
std::const_mem_fun1_ref_t< _Ret, _Tp, _Arg >, 1914
first_argument_type, 1915
result_type, 1915
second_argument_type, 1915
std::const_mem_fun1_t< _Ret, _Tp, _Arg >, 1916
first_argument_type, 1916
result_type, 1916
second_argument_type, 1916
std::const_mem_fun_ref_t< _Ret, _Tp >, 1917
argument_type, 1917
result_type, 1917
std::const_mem_fun_t< _Ret, _Tp >, 1918
argument_type, 1918
result_type, 1918
std::constructible_from, 734
std::contiguous_iterator_tag, 1937
std::convertible_to, 734
std::copy_constructible, 735
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1938
copyable_function, 1939, 1940
operator bool, 1941
operator(), 1941
operator=, 1941
operator==, 1942
swap, 1942
std::counted_iterator< _It >, 1942
std::ctype< _CharT >, 1944
do_is, 1946
do_narrow, 1946, 1947
do_scan_is, 1948
do_scan_not, 1948
do_tolower, 1949
do_toupper, 1949, 1950
do_widen, 1950, 1951
id, 1957
is, 1951, 1952
narrow, 1952, 1953
scan_is, 1953
scan_not, 1954
tolower, 1954, 1955
toupper, 1955
widen, 1956
std::ctype< char >, 1957
~ctype, 1961
char_type, 1960
classic_table, 1961
ctype, 1961
do_is, 1961, 1962
do_narrow, 1963, 1964
do_scan_is, 1965
do_scan_not, 1966
do_tolower, 1967, 1968
do_toupper, 1969, 1970
do_widen, 1970, 1971
id, 1987
is, 1972–1974
narrow, 1975–1977
scan_is, 1978

scan_not, 1979, 1980
table, 1980
table_size, 1988
tolower, 1980–1982
toupper, 1983–1985
widen, 1985–1987
std::ctype< wchar_t >, 1988
~ctype, 1990
char_type, 1990
ctype, 1990
do_is, 1991
do_narrow, 1991, 1992
do_scan_is, 1992
do_scan_not, 1993
do_tolower, 1993, 1994
do_toupper, 1994, 1995
do_widen, 1995
id, 2001
is, 1996
narrow, 1997
scan_is, 1998
scan_not, 1998
tolower, 1999
toupper, 1999, 2000
widen, 2000, 2001
std::ctype_base, 2001
std::ctype_byname< _CharT >, 2002
do_is, 2005
do_narrow, 2005, 2006
do_scan_is, 2007
do_scan_not, 2007
do_tolower, 2008
do_toupper, 2008, 2009
do_widen, 2009, 2010
id, 2016
is, 2010, 2011
narrow, 2011, 2012
scan_is, 2012
scan_not, 2013
tolower, 2013, 2014
toupper, 2014
widen, 2015
std::ctype_byname< char >, 2016
classic_table, 2019
do_is, 2019, 2020
do_narrow, 2021, 2022
do_scan_is, 2023
do_scan_not, 2023, 2024
do_tolower, 2024–2026
do_toupper, 2026, 2027
do_widen, 2028, 2029
id, 2039
is, 2029, 2030
narrow, 2031, 2032
scan_is, 2033
scan_not, 2034
table, 2035
table_size, 2039
tolower, 2035, 2036
toupper, 2036–2038
widen, 2038, 2039
std::decimal, 686
decimal32_to_long_long, 695
std::decimal::decimal128, 2041
decimal128, 2042
std::decimal::decimal32, 2042
decimal32, 2044
std::decimal::decimal64, 2044
decimal64, 2045
std::default_delete< _Tp >, 2046
default_delete, 2047
operator(), 2047
std::default_delete< _Tp[]>, 2047
default_delete, 2048
operator(), 2048
std::default_initializable, 735
std::default_sentinel_t, 2051
std::defer_lock_t, 2052
std::deque< _Tp, _Alloc >, 2055
_M_fill_initialize, 2063
_M_new_elements_at_back, 2063
_M_new_elements_at_front, 2063
_M_pop_back_aux, 2063
_M_pop_front_aux, 2063
_M_push_back_aux, 2063
_M_push_front_aux, 2064
_M_range_check, 2064
_M_range_initialize, 2064
_M_reallocate_map, 2065
_M_reserve_elements_at_back, 2065
_M_reserve_elements_at_front, 2065
_M_reserve_map_at_back, 2065
_M_reserve_map_at_front, 2065
~deque, 2062
assign, 2066
at, 2067
back, 2068
begin, 2068
cbegin, 2068
cend, 2068
clear, 2068
crbegin, 2069
crend, 2069
deque, 2060–2062
emplace, 2069
empty, 2069
end, 2069, 2070
erase, 2070

front, 2070, 2071
get_allocator, 2071
insert, 2071–2073
max_size, 2073
operator=, 2073, 2074
operator[], 2074
pop_back, 2075
pop_front, 2075
push_back, 2075
push_front, 2075
rbegin, 2076
rend, 2076
resize, 2076
shrink_to_fit, 2077
size, 2077
swap, 2077
std::derived_from, 735
std::destroying_delete_t, 2077
std::destructible, 735
std::discard_block_engine< _RandomNumberEngine, __p, __r >, 2083
base, 2085
discard, 2085
discard_block_engine, 2084, 2085
max, 2085
min, 2085
operator<<, 2086
operator>>, 2087
operator(), 2086
operator==, 2087
result_type, 2084
seed, 2086
std::discrete_distribution< _IntType >, 2087
max, 2089
min, 2089
operator<<, 2090
operator>>, 2090
operator(), 2089
operator==, 2090
param, 2089
probabilities, 2089
reset, 2090
result_type, 2089
std::discrete_distribution< _IntType >::param_type, 2728
std::divides< _Tp >, 2091
first_argument_type, 2092
result_type, 2092
second_argument_type, 2092
std::divides< void >, 2092
first_argument_type, 2093
result_type, 2093
second_argument_type, 2093
std::domain_error, 2093
what, 2094
std::enable_if< bool, _Tp >, 2111
std::enable_shared_from_this< _Tp >, 2111
std::encoding_state, 2132
std::equal_to< _Tp >, 2137
first_argument_type, 2137
result_type, 2137
second_argument_type, 2137
std::equal_to< void >, 2138
first_argument_type, 2139
result_type, 2139
second_argument_type, 2139
std::equivalence_relation, 736
std::error_category, 2139
default_error_condition, 2140
equivalent, 2140
message, 2140
name, 2140
operator<=>, 2141
operator==, 2141
std::error_code, 2141
category, 2142
default_error_condition, 2142
error_code, 2142
message, 2142
operator bool, 2142
value, 2142
std::error_condition, 2143
assign, 2144
category, 2144
clear, 2144
error_condition, 2143
message, 2144
operator bool, 2144
value, 2144
std::exception, 2146
what, 2147
std::exception_ptr, 2148
std::experimental, 695
gcd, 706
get_deleter, 706
is_bind_expression_v, 707
is_placeholder_v, 707
lcm, 706
make_boyer_moore_horspool_searcher, 706
make_boyer_moore_searcher, 706
make_default_searcher, 706
make_ostream_joiner, 707
not_fn, 707
sample, 707
std::experimental::filesystem::v1::basic_string_view< _CharT, _Traits >, 1689
std::experimental::filesystem::v1::filesystem_error, 2159
what, 2160
std::experimental::filesystem::v1::path, 2742

std::experimental::filesystem::v1::path::iterator, 2329
std::experimental::filesystem::v1::space_info, 2927
std::experimental::fundamentals_v1::any, 1075
~any, 1077
any, 1076, 1077
clear, 1077
empty, 1077
operator=, 1077, 1078
swap, 1078
type, 1078
std::experimental::fundamentals_v1::bad_any_cast, 1101
what, 1102
std::experimental::fundamentals_v1::bad_optional_access, 1104
what, 1105
std::experimental::fundamentals_v1::basic_string_view< _CharT, _Traits >, 1691
std::experimental::fundamentals_v1::in_place_t, 2271
std::experimental::fundamentals_v1::nullopt_t, 2567
std::experimental::fundamentals_v1::optional< _Tp >, 2688
std::experimental::fundamentals_v2::ostream_joiner< _DelimT, _CharT, _Traits >, 2692
std::experimental::fundamentals_v2::owner_less< shared_ptr< _Tp > >, 2707
first_argument_type, 2707
result_type, 2707
second_argument_type, 2707
std::experimental::fundamentals_v2::owner_less< weak_ptr< _Tp > >, 2709
first_argument_type, 2709
result_type, 2709
second_argument_type, 2710
std::experimental::fundamentals_v2::propagate_const< _Tp >, 2784
std::exponential_distribution< _RealType >, 2148
exponential_distribution, 2150
lambda, 2150
max, 2150
min, 2150
operator(), 2150
operator==, 2151
param, 2150, 2151
reset, 2151
result_type, 2150
std::exponential_distribution< _RealType >::param_type, 2729
std::extent< typename, _Uint >, 2151
std::extreme_value_distribution< _RealType >, 2152
a, 2153
b, 2153
max, 2154
min, 2154
operator(), 2154
operator==, 2155
param, 2154
reset, 2154
result_type, 2153
std::extreme_value_distribution< _RealType >::param_type, 2729
std::filesystem, 708
std::filesystem::directory_entry, 2080
std::filesystem::directory_iterator, 2081
std::filesystem::file_status, 2159
std::filesystem::filesystem_error, 2161
what, 2162
std::filesystem::path, 2745
format, 2747
operator<<, 2748
operator<=>, 2748
operator>>, 2748
operator/, 2748
operator==, 2748
std::filesystem::path::iterator, 2330
std::filesystem::recursive_directory_iterator, 2819
std::filesystem::space_info, 2927
std::fisher_f_distribution< _RealType >, 2162
max, 2164
min, 2164
operator<<, 2165
operator>>, 2165
operator(), 2164
operator==, 2165
param, 2164
reset, 2164
result_type, 2164
std::fisher_f_distribution< _RealType >::param_type, 2730
std::forward_iterator_tag, 2166
std::forward_list< _Tp, _Alloc >, 2170
~forward_list, 2175
assign, 2175, 2176
before_begin, 2176
begin, 2177
cbefore_begin, 2177
cbegin, 2177
cend, 2177
clear, 2177
emplace_after, 2178
emplace_front, 2178
empty, 2178
end, 2178, 2179
erase_after, 2179
forward_list, 2172–2175
front, 2180
get_allocator, 2180
insert_after, 2180, 2181
max_size, 2182

merge, 2182
operator=, 2182, 2183
pop_front, 2183
push_front, 2183
remove, 2184
remove_if, 2184
resize, 2184, 2185
reverse, 2185
sort, 2185
splice_after, 2185–2187
swap, 2187
unique, 2187, 2188
std::fpos< _StateT >, 2188
fpos, 2189
operator streamoff, 2189
operator+, 2189
operator+=, 2189
operator-, 2189
operator-=, 2189
state, 2189, 2190
std::from_chars_result, 2191
std::front_insert_iterator< _Container >, 2192
container_type, 2193
front_insert_iterator, 2193
iterator_category, 2193
operator++, 2193, 2194
operator=, 2194
operator*, 2193
pointer, 2193
reference, 2193
value_type, 2193
std::function< _Res(_ArgTypes...)>, 2194
function, 2195, 2196
operator bool, 2197
operator(), 2197
operator=, 2197–2199
swap, 2199
target, 2199, 2200
target_type, 2200
std::function_ref< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2200
function_ref, 2201, 2202
operator(), 2202
std::future< _Res >, 2202
_M_get_result, 2204
future, 2204
get, 2204
std::future< _Res & >, 2204
_M_get_result, 2205
future, 2205
get, 2205
std::future< void >, 2206
_M_get_result, 2207
future, 2207
get, 2207
std::future_error, 2207
what, 2208
std::gamma_distribution< _RealType >, 2208
alpha, 2210
beta, 2210
gamma_distribution, 2210
max, 2210
min, 2210
operator<<, 2211
operator>>, 2212
operator(), 2210, 2211
operator==, 2212
param, 2211
reset, 2211
result_type, 2210
std::gamma_distribution< _RealType >::param_type, 2730
std::geometric_distribution< _IntType >, 2212
max, 2214
min, 2214
operator(), 2214
operator==, 2215
p, 2214
param, 2214
reset, 2214
result_type, 2213
std::geometric_distribution< _IntType >::param_type, 2731
std::greater< _Tp >, 2227
first_argument_type, 2227
result_type, 2227
second_argument_type, 2227
std::greater< void >, 2228
first_argument_type, 2229
result_type, 2229
second_argument_type, 2229
std::greater_equal< _Tp >, 2229
first_argument_type, 2230
result_type, 2230
second_argument_type, 2230
std::greater_equal< void >, 2230
first_argument_type, 2231
result_type, 2231
second_argument_type, 2231
std::gslice, 2232
std::gslice_array< _Tp >, 2233
operator<<=, 2235
operator>>=, 2235
operator+=, 2235
operator-=, 2235
operator/=, 2235
operator%=, 2234
operator&=, 2234

operator*=, 2235
operator^=, 2235
operator|=, 2235
std::has_virtual_destructor< _Tp >, 2236
std::hash< __debug::bitset< _Nb > >, 2236
std::hash< __debug::vector< bool, _Alloc > >, 2237
std::hash< __gnu_cxx::__u16vstring >, 2237
std::hash< __gnu_cxx::__u32vstring >, 2237
std::hash< __gnu_cxx::__vstring >, 2237
std::hash< __gnu_cxx::__wvstring >, 2238
std::hash< __gnu_cxx::throw_value_limit >, 2238
argument_type, 2239
result_type, 2239
std::hash< __gnu_cxx::throw_value_random >, 2239
argument_type, 2240
result_type, 2240
std::hash< __gnu_debug::basic_string< _CharT > >, 2240
std::hash< __shared_ptr< _Tp, _Lp > >, 2240
std::hash< _Tp >, 2236
std::hash< _Tp * >, 2241
std::hash< basic_string< char, char_traits< char >, _Alloc > >, 2241
std::hash< basic_string< char16_t, char_traits< char16_t >, _Alloc > >, 2241
std::hash< basic_string< char32_t, char_traits< char32_t >, _Alloc > >, 2242
std::hash< basic_string< wchar_t, char_traits< wchar_t >, _Alloc > >, 2242
std::hash< bool >, 2242
std::hash< char >, 2243
std::hash< char16_t >, 2243
std::hash< char32_t >, 2243
std::hash< double >, 2243
std::hash< error_code >, 2244
std::hash< error_condition >, 2244
std::hash< experimental::optional< _Tp > >, 2244
std::hash< experimental::shared_ptr< _Tp > >, 2245
std::hash< float >, 2245
std::hash< int >, 2245
std::hash< long >, 2246
std::hash< long double >, 2246
std::hash< long long >, 2246
std::hash< shared_ptr< _Tp > >, 2246
std::hash< short >, 2247
std::hash< signed char >, 2247
std::hash< thread::id >, 2247
std::hash< type_index >, 2247
std::hash< unique_ptr< _Tp, _Dp > >, 2248
std::hash< unsigned char >, 2248
std::hash< unsigned int >, 2248
std::hash< unsigned long >, 2249
std::hash< unsigned long long >, 2249
std::hash< unsigned short >, 2249
std::hash< wchar_t >, 2249
std::hash<::bitset< _Nb > >, 2250
std::hash<::vector< bool, _Alloc > >, 2250
std::identity, 2271
std::independent_bits_engine< _RandomNumberEngine, __w, _UIntType >, 2271
base, 2273
discard, 2273
independent_bits_engine, 2272, 2273
max, 2274
min, 2274
operator>>, 2275
operator(), 2274
operator==, 2275
result_type, 2272
seed, 2274
std::indirect_array< _Tp >, 2275
operator<<=, 2277
operator>>=, 2278
operator+=, 2277
operator-=, 2277
operator/=, 2277
operator%=, 2277
operator&=, 2277
operator*=, 2277
operator^=, 2278
operator|=, 2278
std::indirectly_comparable, 736
std::indirectly_copyable, 736
std::indirectly_movable, 736
std::indirectly_readable, 736
std::indirectly_swappable, 737
std::indirectly_writable, 737
std::initializer_list< _E >, 2278
std::inout_ptr_t< _Smart, _Pointer, _Args >, 2279
std::input_iterator_tag, 2279
std::insert_iterator< _Container >, 2281
container_type, 2283
insert_iterator, 2283
iterator_category, 2283
operator++, 2283
operator=, 2284
operator*, 2283
pointer, 2283
reference, 2283
value_type, 2283
std::integer_sequence< _Tp, _Idx >, 2284
std::integral_constant< _Tp, __v >, 2285
std::invalid_argument, 2285
what, 2286
std::invocable, 737
std::ios_base, 2286
_M_getloc, 2292
~ios_base, 2292

adjustfield, 2297
app, 2297
ate, 2297
badbit, 2297
basefield, 2297
beg, 2297
binary, 2298
boolalpha, 2298
cur, 2298
dec, 2298
end, 2298
eofbit, 2298
event, 2291
event_callback, 2290
failbit, 2298
fixed, 2299
flags, 2292
floatfield, 2299
fmtflags, 2290
getloc, 2292
goodbit, 2299
hex, 2299
imbue, 2293
in, 2299
internal, 2300
iostate, 2291
iword, 2293
left, 2300
oct, 2300
openmode, 2291
out, 2300
precision, 2293, 2294
pword, 2294
register_callback, 2294
right, 2300
scientific, 2300
seekdir, 2291
setf, 2295
showbase, 2301
showpoint, 2301
showpos, 2301
skipws, 2301
sync_with_stdio, 2295
trunc, 2301
unitbuf, 2301
unsetf, 2296
uppercase, 2301
width, 2296
xalloc, 2296
std::ios_base::failure, 2158
what, 2158
std::is_abstract< _Tp >, 2301
std::is_arithmetic< _Tp >, 2302
std::is_array< _Tp >, 2302
std::is_assignable< _Tp, _Up >, 2302
std::is_base_of< _Base, _Derived >, 2302
std::is_bind_expression< _Bind< _Signature > >, 2303
std::is_bind_expression< _Bind_result< _Result, _Signature > >, 2303
std::is_bind_expression< _Tp >, 2303
std::is_bind_expression< const _Bind< _Signature > >, 2304
std::is_bind_expression< const _Bind_result< _Result, _Signature > >, 2304
std::is_bind_expression< const volatile _Bind< _Signature > >, 2304
std::is_bind_expression< const volatile _Bind_result< _Result, _Signature > >, 2304
std::is_bind_expression< volatile _Bind< _Signature > >, 2305
std::is_bind_expression< volatile _Bind_result< _Result, _Signature > >, 2305
std::is_class< _Tp >, 2305
std::is_compound< _Tp >, 2305
std::is_const< _Tp >, 2306
std::is_constructible< _Tp, _Args >, 2306
std::is_copy_assignable< _Tp >, 2306
std::is_copy_constructible< _Tp >, 2306
std::is_default_constructible< _Tp >, 2306
std::is_destructible< _Tp >, 2307
std::is_empty< _Tp >, 2307
std::is_enum< _Tp >, 2307
std::is_error_code_enum< _Tp >, 2307
std::is_error_code_enum< future_errc >, 2308
std::is_error_condition_enum< _Tp >, 2308
std::is_floating_point< _Tp >, 2308
std::is_function< _Tp >, 2309
std::is_fundamental< _Tp >, 2309
std::is_integral< _Tp >, 2309
std::is_layout_compatible< _Tp, _Up >, 2309
std::is_literal_type< _Tp >, 2310
std::is_lvalue_reference< typename >, 2310
std::is_member_function_pointer< _Tp >, 2310
std::is_member_object_pointer< _Tp >, 2311
std::is_member_pointer< _Tp >, 2311
std::is_move_assignable< _Tp >, 2311
std::is_move_constructible< _Tp >, 2311
std::is_nothrow_assignable< _Tp, _Up >, 2311
std::is_nothrow_constructible< _Tp, _Args >, 2312
std::is_nothrow_copy_assignable< _Tp >, 2312
std::is_nothrow_copy_constructible< _Tp >, 2312
std::is_nothrow_default_constructible< _Tp >, 2312
std::is_nothrow_destructible< _Tp >, 2313
std::is_nothrow_move_assignable< _Tp >, 2313
std::is_nothrow_move_constructible< _Tp >, 2313
std::is_object< _Tp >, 2313
std::is_placeholder< _Placeholder< _Num > >, 2315
std::is_placeholder< _Tp >, 2314

std::is_pod< _Tp >, 2316
std::is_pointer< _Tp >, 2316
std::is_pointer_interconvertible_base_of< _Base, _Derived >, 2316
std::is_polymorphic< _Tp >, 2316
std::is_reference< _Tp >, 2317
std::is_rvalue_reference< typename >, 2317
std::is_same< _Tp, _Up >, 2317
std::is_scalar< _Tp >, 2317
std::is_signed< _Tp >, 2318
std::is_standard_layout< _Tp >, 2318
std::is_trivial< _Tp >, 2318
std::is_trivially_assignable< _Tp, _Up >, 2318
std::is_trivially_constructible< _Tp, _Args >, 2319
std::is_trivially_copy_assignable< _Tp >, 2319
std::is_trivially_copy_constructible< _Tp >, 2319
std::is_trivially_copyable< _Tp >, 2319
std::is_trivially_default_constructible< _Tp >, 2320
std::is_trivially_destructible< _Tp >, 2320
std::is_trivially_move_assignable< _Tp >, 2320
std::is_trivially_move_constructible< _Tp >, 2320
std::is_union< _Tp >, 2321
std::is_unsigned< _Tp >, 2321
std::is_void< _Tp >, 2321
std::is_volatile< _Tp >, 2321
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2322
  difference_type, 2323
  istream_iterator, 2323, 2324
  iterator_category, 2323
  operator==, 2324
  pointer, 2323
  reference, 2323
  value_type, 2323
std::istreambuf_iterator< _CharT, _Traits >, 2324
  char_type, 2326
  difference_type, 2326
  equal, 2327
  int_type, 2326
  istream_type, 2326
  istreambuf_iterator, 2327
  iterator_category, 2326
  operator++, 2327
  operator*, 2327
  pointer, 2326
  reference, 2326
  streambuf_type, 2326
  traits_type, 2326
  value_type, 2326
std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, 2330
  difference_type, 2331
  iterator_category, 2331
  pointer, 2332
  reference, 2332
  value_type, 2332
std::iterator_traits< _Iterator >, 2332
std::iterator_traits< _Tp * >, 2332
std::length_error, 2343
  what, 2343
std::less< _Tp >, 2344
  first_argument_type, 2344
  result_type, 2344
  second_argument_type, 2344
std::less< void >, 2345
  first_argument_type, 2346
  result_type, 2346
  second_argument_type, 2346
std::less_equal< _Tp >, 2346
  first_argument_type, 2347
  result_type, 2347
  second_argument_type, 2347
std::less_equal< void >, 2347
  first_argument_type, 2348
  result_type, 2348
  second_argument_type, 2348
std::linear_congruential_engine< _UIntType, __a, __c, __m >, 2349
  discard, 2351
  increment, 2354
  linear_congruential_engine, 2351
  max, 2351
  min, 2352
  modulus, 2354
  multiplier, 2354
  operator<<, 2353
  operator>>, 2353
  operator(), 2352
  operator==, 2353
  result_type, 2351
  seed, 2352
std::list< _Tp, _Alloc >, 2357
  _M_create_node, 2364
  ~list, 2364
  assign, 2364, 2365
  back, 2365
  begin, 2365, 2366
  cbegin, 2366
  cend, 2366
  clear, 2366
  crbegin, 2366
  crend, 2366
  emplace, 2366
  empty, 2367
  end, 2367
  erase, 2367, 2368
  front, 2368
  get_allocator, 2368
  insert, 2368–2370

list, 2362, 2363
max_size, 2371
merge, 2371
operator=, 2372
pop_back, 2372
pop_front, 2373
push_back, 2373
push_front, 2373
rbegin, 2373, 2374
remove, 2374
remove_if, 2374
rend, 2374
resize, 2375
reverse, 2375
size, 2375
sort, 2375, 2376
splice, 2376, 2377
swap, 2378
unique, 2378
std::literals, 711
std::literals::chrono_literals, 712
std::locale, 2380
~locale, 2385
all, 2388
category, 2382
classic, 2385
collate, 2388
combine, 2385
ctype, 2388
global, 2385
has_facet, 2387
locale, 2382–2384
messages, 2389
monetary, 2389
name, 2386
none, 2389
numeric, 2389
operator(), 2386
operator=, 2386
operator==, 2387
time, 2389
use_facet, 2388
std::locale::facet, 2155
~facet, 2157
facet, 2157
std::locale::id, 2268
has_facet, 2269
id, 2269
use_facet, 2269
std::lock_guard< _Mutex >, 2389
std::logic_error, 2390
logic_error, 2391
what, 2391
std::logical_and< _Tp >, 2391
first_argument_type, 2392
result_type, 2392
second_argument_type, 2392
std::logical_and< void >, 2393
first_argument_type, 2393
result_type, 2393
second_argument_type, 2393
std::logical_not< _Tp >, 2394
argument_type, 2394
result_type, 2394
std::logical_not< void >, 2395
argument_type, 2395
result_type, 2395
std::logical_or< _Tp >, 2396
first_argument_type, 2396
result_type, 2396
second_argument_type, 2396
std::logical_or< void >, 2397
first_argument_type, 2397
result_type, 2397
second_argument_type, 2397
std::lognormal_distribution< _RealType >, 2398
max, 2399
min, 2399
operator<<, 2400
operator>>, 2400
operator(), 2399
operator==, 2400
param, 2399
reset, 2400
result_type, 2399
std::lognormal_distribution< _RealType >::param_type, 2731
std::make_signed< _Tp >, 2408
std::make_unsigned< _Tp >, 2408
std::map< _Key, _Tp, _Compare, _Alloc >, 2411
~map, 2418
at, 2418
begin, 2418, 2419
cbegin, 2419
cend, 2419
clear, 2419
contains, 2419
count, 2420
crbegin, 2420
crend, 2420
emplace, 2420
emplace_hint, 2421
empty, 2421
end, 2422
equal_range, 2422
erase, 2423, 2424
find, 2425
get_allocator, 2426

insert, 2426–2429
key_comp, 2429
lower_bound, 2429, 2430
map, 2415–2417
max_size, 2430
operator=, 2430, 2431
operator[], 2431
rbegin, 2431
rend, 2432
size, 2432
swap, 2432
upper_bound, 2432, 2433
value_comp, 2433
std::mask_array< _Tp >, 2433
operator<<=, 2435
operator>>=, 2435
operator+=, 2435
operator-=, 2435
operator/=, 2435
operator%=, 2435
operator&=, 2435
operator*=, 2435
operator^=, 2436
operator|=, 2436
std::match_results< _Bi_iter, _Alloc >, 2437
~match_results, 2440
begin, 2441
cbegin, 2441
cend, 2441
empty, 2441
end, 2441
format, 2442
get_allocator, 2442
length, 2443
match_results, 2440
max_size, 2443
operator=, 2443
operator[], 2443
position, 2444
prefix, 2444
ready, 2444
size, 2445
str, 2445
suffix, 2445
swap, 2446
std::mem_fun1_ref_t< _Ret, _Tp, _Arg >, 2448
first_argument_type, 2449
result_type, 2449
second_argument_type, 2449
std::mem_fun1_t< _Ret, _Tp, _Arg >, 2450
first_argument_type, 2450
result_type, 2450
second_argument_type, 2450
std::mem_fun_ref_t< _Ret, _Tp >, 2451
argument_type, 2451
result_type, 2451
std::mem_fun_t< _Ret, _Tp >, 2452
argument_type, 2452
result_type, 2452
std::mergeable, 738
std::mersenne_twister_engine< _UIntType, __w, __n, __m, __r, __a, __u, __d, __s, __b, __t, __c, __l, __f >, 2453
discard, 2456
max, 2456
mersenne_twister_engine, 2456
min, 2456
operator<<, 2457
operator>>, 2458
operator==, 2457
result_type, 2456
std::messages< _CharT >, 2458
~messages, 2461
char_type, 2460
do_get, 2461
id, 2461
messages, 2460, 2461
string_type, 2460
std::messages_base, 2462
std::messages_byname< _CharT >, 2462
do_get, 2464
id, 2464
std::minus< _Tp >, 2464
first_argument_type, 2465
result_type, 2465
second_argument_type, 2465
std::minus< void >, 2466
first_argument_type, 2466
result_type, 2466
second_argument_type, 2466
std::modulus< _Tp >, 2467
first_argument_type, 2468
result_type, 2468
second_argument_type, 2468
std::modulus< void >, 2468
first_argument_type, 2469
result_type, 2469
second_argument_type, 2469
std::money_base, 2469
std::money_get< _CharT, _InIter >, 2471
~money_get, 2472
char_type, 2472
do_get, 2473
get, 2473, 2474
id, 2475
iter_type, 2472
money_get, 2472
string_type, 2472

std::money_put< _CharT, _OutIter >, 2475
~money_put, 2477
char_type, 2476
do_put, 2477, 2478
id, 2480
iter_type, 2476
money_put, 2477
put, 2478, 2479
string_type, 2476
std::moneypunct< _CharT, _Intl >, 2480
~moneypunct, 2483
char_type, 2482
curr_symbol, 2483
decimal_point, 2483
do_curr_symbol, 2484
do_decimal_point, 2484
do_frac_digits, 2484
do_grouping, 2484
do_neg_format, 2485
do_negative_sign, 2485
do_pos_format, 2485
do_positive_sign, 2486
do_thousands_sep, 2486
frac_digits, 2486
grouping, 2487
id, 2489
intl, 2489
moneypunct, 2482, 2483
neg_format, 2487
negative_sign, 2487
pos_format, 2488
positive_sign, 2488
string_type, 2482
thousands_sep, 2488
std::moneypunct_byname< _CharT, _Intl >, 2489
curr_symbol, 2491
decimal_point, 2491
do_curr_symbol, 2491
do_decimal_point, 2491
do_frac_digits, 2492
do_grouping, 2492
do_neg_format, 2492
do_negative_sign, 2493
do_pos_format, 2493
do_positive_sign, 2493
do_thousands_sep, 2494
frac_digits, 2494
grouping, 2494
id, 2496
neg_format, 2494
negative_sign, 2495
pos_format, 2495
positive_sign, 2496
thousands_sep, 2496
std::move_constructible, 738
std::move_iterator< _Iterator >, 2498
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2499
move_only_function, 2500, 2501
operator bool, 2502
operator(), 2502
operator=, 2502
operator==, 2503
swap, 2502, 2503
std::move_sentinel< _Sent >, 2503
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2506
~multimap, 2512
begin, 2512
cbegin, 2512
cend, 2512
clear, 2513
contains, 2513
count, 2513
crbegin, 2514
crend, 2514
emplace, 2514
emplace_hint, 2515
empty, 2515
end, 2515
equal_range, 2516
erase, 2516–2518
find, 2518
get_allocator, 2519
insert, 2519–2522
key_comp, 2522
lower_bound, 2522, 2523
max_size, 2523
multimap, 2509–2511
operator=, 2523, 2524
rbegin, 2524
rend, 2524
size, 2525
swap, 2525
upper_bound, 2525
value_comp, 2526
std::multiplies< _Tp >, 2526
first_argument_type, 2527
result_type, 2527
second_argument_type, 2527
std::multiplies< void >, 2527
first_argument_type, 2528
result_type, 2528
second_argument_type, 2528
std::multiset< _Key, _Compare, _Alloc >, 2530
~multiset, 2536
begin, 2536
cbegin, 2536
cend, 2536

clear, 2536
contains, 2537
count, 2537
crbegin, 2538
crend, 2538
emplace, 2538
emplace_hint, 2538
empty, 2539
end, 2539
equal_range, 2539
erase, 2540, 2541
find, 2541
get_allocator, 2542
insert, 2542, 2543
key_comp, 2544
lower_bound, 2544
max_size, 2545
multiset, 2533–2536
operator=, 2545
rbegin, 2545
rend, 2546
size, 2546
swap, 2546
upper_bound, 2546, 2547
value_comp, 2547
std::mutex, 2551
std::negate< _Tp >, 2551
argument_type, 2552
result_type, 2552
std::negate< void >, 2552
argument_type, 2553
result_type, 2553
std::negative_binomial_distribution< _IntType >, 2553
k, 2554
max, 2554
min, 2554
operator<<, 2555
operator>>, 2556
operator(), 2554
operator==, 2556
p, 2555
param, 2555
reset, 2555
result_type, 2554
std::negative_binomial_distribution< _IntType >::param_type, 2732
std::nested_exception, 2556
nested_exception, 2557
nested_ptr, 2557
rethrow_nested, 2557
std::normal_distribution< _RealType >, 2559
max, 2561
mean, 2561
min, 2561
normal_distribution, 2561
operator<<, 2562
operator>>, 2563
operator(), 2561
operator==, 2563
param, 2562
reset, 2562
result_type, 2561
stddev, 2562
std::normal_distribution< _RealType >::param_type, 2732
std::not_equal_to< _Tp >, 2563
first_argument_type, 2564
result_type, 2564
second_argument_type, 2564
std::not_equal_to< void >, 2565
first_argument_type, 2566
result_type, 2566
second_argument_type, 2566
std::num_get< _CharT, _InIter >, 2568
~num_get, 2570
char_type, 2570
do_get, 2570–2577
get, 2577–2584
id, 2585
iter_type, 2570
num_get, 2570
std::num_put< _CharT, _OutIter >, 2585
~num_put, 2587
char_type, 2587
do_put, 2588–2591
id, 2598
iter_type, 2587
num_put, 2587
put, 2592–2597
std::numeric_limits< _Tp >, 2598
denorm_min, 2599
digits, 2601
digits10, 2601
epsilon, 2599
has_denorm, 2601
has_denorm_loss, 2601
has_infinity, 2601
has_quiet_NaN, 2601
has_signaling_NaN, 2601
infinity, 2599
is_bounded, 2601
is_exact, 2601
is_iec559, 2601
is_integer, 2601
is_modulo, 2602
is_signed, 2602
is_specialized, 2602
lowest, 2600

max, 2600
max_digits10, 2602
max_exponent, 2602
max_exponent10, 2602
min, 2600
min_exponent, 2602
min_exponent10, 2602
quiet_NaN, 2600
radix, 2602
round_error, 2600
round_style, 2603
signaling_NaN, 2600
tinyness_before, 2603
traps, 2603
std::numeric_limits< bool >, 2603
digits, 2605
digits10, 2605
has_denorm, 2605
has_denorm_loss, 2605
has_infinity, 2605
has_quiet_NaN, 2605
has_signaling_NaN, 2605
is_bounded, 2605
is_exact, 2605
is_iec559, 2605
is_integer, 2605
is_modulo, 2606
is_signed, 2606
is_specialized, 2606
max_digits10, 2606
max_exponent, 2606
max_exponent10, 2606
min_exponent, 2606
min_exponent10, 2606
radix, 2606
round_style, 2606
tinyness_before, 2607
traps, 2607
std::numeric_limits< char >, 2607
digits, 2609
digits10, 2609
has_denorm, 2609
has_denorm_loss, 2609
has_infinity, 2609
has_quiet_NaN, 2609
has_signaling_NaN, 2609
is_bounded, 2609
is_exact, 2609
is_iec559, 2609
is_integer, 2609
is_modulo, 2610
is_signed, 2610
is_specialized, 2610
max_digits10, 2610
max_exponent, 2610
max_exponent10, 2610
min_exponent, 2610
min_exponent10, 2610
radix, 2610
round_style, 2610
tinyness_before, 2611
traps, 2611
std::numeric_limits< char16_t >, 2611
digits, 2613
digits10, 2613
has_denorm, 2613
has_denorm_loss, 2613
has_infinity, 2613
has_quiet_NaN, 2613
has_signaling_NaN, 2613
is_bounded, 2613
is_exact, 2613
is_iec559, 2613
is_integer, 2613
is_modulo, 2614
is_signed, 2614
is_specialized, 2614
max_digits10, 2614
max_exponent, 2614
max_exponent10, 2614
min_exponent, 2614
min_exponent10, 2614
radix, 2614
round_style, 2614
tinyness_before, 2615
traps, 2615
std::numeric_limits< char32_t >, 2615
digits, 2617
digits10, 2617
has_denorm, 2617
has_denorm_loss, 2617
has_infinity, 2617
has_quiet_NaN, 2617
has_signaling_NaN, 2617
is_bounded, 2617
is_exact, 2617
is_iec559, 2617
is_integer, 2617
is_modulo, 2618
is_signed, 2618
is_specialized, 2618
max_digits10, 2618
max_exponent, 2618
max_exponent10, 2618
min_exponent, 2618
min_exponent10, 2618
radix, 2618
round_style, 2618

tinyness_before, 2619
traps, 2619
std::numeric_limits< double >, 2619
digits, 2621
digits10, 2621
has_denorm, 2621
has_denorm_loss, 2621
has_infinity, 2621
has_quiet_NaN, 2621
has_signaling_NaN, 2621
is_bounded, 2621
is_exact, 2621
is_iec559, 2621
is_integer, 2621
is_modulo, 2622
is_signed, 2622
is_specialized, 2622
max_digits10, 2622
max_exponent, 2622
max_exponent10, 2622
min_exponent, 2622
min_exponent10, 2622
radix, 2622
round_style, 2622
tinyness_before, 2623
traps, 2623
std::numeric_limits< float >, 2623
digits, 2625
digits10, 2625
has_denorm, 2625
has_denorm_loss, 2625
has_infinity, 2625
has_quiet_NaN, 2625
has_signaling_NaN, 2625
is_bounded, 2625
is_exact, 2625
is_iec559, 2625
is_integer, 2625
is_modulo, 2626
is_signed, 2626
is_specialized, 2626
max_digits10, 2626
max_exponent, 2626
max_exponent10, 2626
min_exponent, 2626
min_exponent10, 2626
radix, 2626
round_style, 2626
tinyness_before, 2627
traps, 2627
std::numeric_limits< int >, 2627
digits, 2629
digits10, 2629
has_denorm, 2629
has_denorm_loss, 2629
has_infinity, 2629
has_quiet_NaN, 2629
has_signaling_NaN, 2629
is_bounded, 2629
is_exact, 2629
is_iec559, 2629
is_integer, 2629
is_modulo, 2630
is_signed, 2630
is_specialized, 2630
max_digits10, 2630
max_exponent, 2630
max_exponent10, 2630
min_exponent, 2630
min_exponent10, 2630
radix, 2630
round_style, 2630
tinyness_before, 2631
traps, 2631
std::numeric_limits< long >, 2631
digits, 2633
digits10, 2633
has_denorm, 2633
has_denorm_loss, 2633
has_infinity, 2633
has_quiet_NaN, 2633
has_signaling_NaN, 2633
is_bounded, 2633
is_exact, 2633
is_iec559, 2633
is_integer, 2633
is_modulo, 2634
is_signed, 2634
is_specialized, 2634
max_digits10, 2634
max_exponent, 2634
max_exponent10, 2634
min_exponent, 2634
min_exponent10, 2634
radix, 2634
round_style, 2634
tinyness_before, 2635
traps, 2635
std::numeric_limits< long double >, 2635
digits, 2637
digits10, 2637
has_denorm, 2637
has_denorm_loss, 2637
has_infinity, 2637
has_quiet_NaN, 2637
has_signaling_NaN, 2637
is_bounded, 2637
is_exact, 2637

is_iec559, 2637
is_integer, 2637
is_modulo, 2638
is_signed, 2638
is_specialized, 2638
max_digits10, 2638
max_exponent, 2638
max_exponent10, 2638
min_exponent, 2638
min_exponent10, 2638
radix, 2638
round_style, 2638
tinyness_before, 2639
traps, 2639
std::numeric_limits< long long >, 2639
digits, 2641
digits10, 2641
has_denorm, 2641
has_denorm_loss, 2641
has_infinity, 2641
has_quiet_NaN, 2641
has_signaling_NaN, 2641
is_bounded, 2641
is_exact, 2641
is_iec559, 2641
is_integer, 2641
is_modulo, 2642
is_signed, 2642
is_specialized, 2642
max_digits10, 2642
max_exponent, 2642
max_exponent10, 2642
min_exponent, 2642
min_exponent10, 2642
radix, 2642
round_style, 2642
tinyness_before, 2643
traps, 2643
std::numeric_limits< short >, 2643
digits, 2645
digits10, 2645
has_denorm, 2645
has_denorm_loss, 2645
has_infinity, 2645
has_quiet_NaN, 2645
has_signaling_NaN, 2645
is_bounded, 2645
is_exact, 2645
is_iec559, 2645
is_integer, 2645
is_modulo, 2646
is_signed, 2646
is_specialized, 2646
max_digits10, 2646
max_exponent, 2646
max_exponent10, 2646
min_exponent, 2646
min_exponent10, 2646
radix, 2646
round_style, 2646
tinyness_before, 2647
traps, 2647
std::numeric_limits< signed char >, 2647
digits, 2649
digits10, 2649
has_denorm, 2649
has_denorm_loss, 2649
has_infinity, 2649
has_quiet_NaN, 2649
has_signaling_NaN, 2649
is_bounded, 2649
is_exact, 2649
is_iec559, 2649
is_integer, 2649
is_modulo, 2650
is_signed, 2650
is_specialized, 2650
max_digits10, 2650
max_exponent, 2650
max_exponent10, 2650
min_exponent, 2650
min_exponent10, 2650
radix, 2650
round_style, 2650
tinyness_before, 2651
traps, 2651
std::numeric_limits< unsigned char >, 2651
digits, 2653
digits10, 2653
has_denorm, 2653
has_denorm_loss, 2653
has_infinity, 2653
has_quiet_NaN, 2653
has_signaling_NaN, 2653
is_bounded, 2653
is_exact, 2653
is_iec559, 2653
is_integer, 2653
is_modulo, 2654
is_signed, 2654
is_specialized, 2654
max_digits10, 2654
max_exponent, 2654
max_exponent10, 2654
min_exponent, 2654
min_exponent10, 2654
radix, 2654
round_style, 2654

tinyness_before, 2655
traps, 2655
std::numeric_limits< unsigned int >, 2655
digits, 2657
digits10, 2657
has_denorm, 2657
has_denorm_loss, 2657
has_infinity, 2657
has_quiet_NaN, 2657
has_signaling_NaN, 2657
is_bounded, 2657
is_exact, 2657
is_iec559, 2657
is_integer, 2657
is_modulo, 2658
is_signed, 2658
is_specialized, 2658
max_digits10, 2658
max_exponent, 2658
max_exponent10, 2658
min_exponent, 2658
min_exponent10, 2658
radix, 2658
round_style, 2658
tinyness_before, 2659
traps, 2659
std::numeric_limits< unsigned long >, 2659
digits, 2661
digits10, 2661
has_denorm, 2661
has_denorm_loss, 2661
has_infinity, 2661
has_quiet_NaN, 2661
has_signaling_NaN, 2661
is_bounded, 2661
is_exact, 2661
is_iec559, 2661
is_integer, 2661
is_modulo, 2662
is_signed, 2662
is_specialized, 2662
max_digits10, 2662
max_exponent, 2662
max_exponent10, 2662
min_exponent, 2662
min_exponent10, 2662
radix, 2662
round_style, 2662
tinyness_before, 2663
traps, 2663
std::numeric_limits< unsigned long long >, 2663
digits, 2665
digits10, 2665
has_denorm, 2665
has_denorm_loss, 2665
has_infinity, 2665
has_quiet_NaN, 2665
has_signaling_NaN, 2665
is_bounded, 2665
is_exact, 2665
is_iec559, 2665
is_integer, 2665
is_modulo, 2666
is_signed, 2666
is_specialized, 2666
max_digits10, 2666
max_exponent, 2666
max_exponent10, 2666
min_exponent, 2666
min_exponent10, 2666
radix, 2666
round_style, 2666
tinyness_before, 2667
traps, 2667
std::numeric_limits< unsigned short >, 2667
digits, 2669
digits10, 2669
has_denorm, 2669
has_denorm_loss, 2669
has_infinity, 2669
has_quiet_NaN, 2669
has_signaling_NaN, 2669
is_bounded, 2669
is_exact, 2669
is_iec559, 2669
is_integer, 2669
is_modulo, 2670
is_signed, 2670
is_specialized, 2670
max_digits10, 2670
max_exponent, 2670
max_exponent10, 2670
min_exponent, 2670
min_exponent10, 2670
radix, 2670
round_style, 2670
tinyness_before, 2671
traps, 2671
std::numeric_limits< wchar_t >, 2671
digits, 2673
digits10, 2673
has_denorm, 2673
has_denorm_loss, 2673
has_infinity, 2673
has_quiet_NaN, 2673
has_signaling_NaN, 2673
is_bounded, 2673
is_exact, 2673

is_iec559, 2673
is_integer, 2673
is_modulo, 2674
is_signed, 2674
is_specialized, 2674
max_digits10, 2674
max_exponent, 2674
max_exponent10, 2674
min_exponent, 2674
min_exponent10, 2674
radix, 2674
round_style, 2674
tinyness_before, 2675
traps, 2675
std::numpunct< _CharT >, 2675
~numpunct, 2678
char_type, 2677
decimal_point, 2678
do_decimal_point, 2678
do_falsename, 2678
do_grouping, 2678
do_thousands_sep, 2679
do_truename, 2679
falsename, 2679
grouping, 2679
id, 2680
numpunct, 2677
string_type, 2677
thousands_sep, 2680
truename, 2680
std::numpunct_byname< _CharT >, 2681
decimal_point, 2682
do_decimal_point, 2682
do_falsename, 2682
do_grouping, 2683
do_thousands_sep, 2683
do_truename, 2683
falsename, 2683
grouping, 2684
id, 2685
thousands_sep, 2684
truename, 2684
std::once_flag, 2687
call_once, 2687
once_flag, 2687
operator=, 2687
std::ostream_iterator< _Tp, _CharT, _Traits >, 2689
char_type, 2690
difference_type, 2690
iterator_category, 2690
operator=, 2692
ostream_iterator, 2691, 2692
ostream_type, 2690
pointer, 2691
reference, 2691
traits_type, 2691
value_type, 2691
std::ostreambuf_iterator< _CharT, _Traits >, 2693
char_type, 2694
difference_type, 2694
failed, 2695
iterator_category, 2694
operator++, 2695
operator=, 2696
operator*, 2695
ostream_type, 2694
ostreambuf_iterator, 2695
pointer, 2694
reference, 2694
streambuf_type, 2694
traits_type, 2695
value_type, 2695
std::out_of_range, 2696
what, 2697
std::out_ptr_t< _Smart, _Pointer, _Args >, 2697
std::output_iterator_tag, 2698
std::overflow_error, 2705
what, 2706
std::owner_less< _Tp >, 2707
std::owner_less< shared_ptr< _Tp > >, 2708
first_argument_type, 2708
result_type, 2708
second_argument_type, 2708
std::owner_less< void >, 2708
first_argument_type, 2709
result_type, 2709
second_argument_type, 2709
std::owner_less< weak_ptr< _Tp > >, 2710
first_argument_type, 2710
result_type, 2710
second_argument_type, 2710
std::packaged_task< _Res(_ArgTypes...)>, 2711
std::pair< _T1, _T2 >, 2716
explicit, 2719, 2720
first, 2721
first_type, 2719
operator=, 2720, 2721
pair, 2719
second, 2722
second_type, 2719
swap, 2721
std::permutable, 738
std::philox_engine< _UIntType, __w, __n, __r, __consts >, 2748
discard, 2750
max, 2750
min, 2750
operator<<, 2751

operator>>, 2752
operator(), 2751
operator==, 2752
philox_engine, 2750
seed, 2751
set_counter, 2751
std::piecewise_constant_distribution< _RealType >, 2752
densities, 2754
intervals, 2754
max, 2754
min, 2754
operator<<, 2755
operator>>, 2755
operator(), 2754
operator==, 2755
param, 2754, 2755
reset, 2755
result_type, 2754
std::piecewise_constant_distribution< _RealType >::param_type, 2733
std::piecewise_construct_t, 2756
std::piecewise_linear_distribution< _RealType >, 2756
densities, 2758
intervals, 2758
max, 2758
min, 2758
operator<<, 2759
operator>>, 2759
operator(), 2758
operator==, 2759
param, 2758
reset, 2758
result_type, 2757
std::piecewise_linear_distribution< _RealType >::param_type, 2734
std::placeholders, 713
std::plus< _Tp >, 2760
first_argument_type, 2760
operator(), 2761
result_type, 2760
second_argument_type, 2761
std::pmr::memory_resource, 2453
std::pmr::monotonic_buffer_resource, 2496
do_allocate, 2498
do_deallocate, 2498
do_is_equal, 2498
std::pmr::polymorphic_allocator< _Tp >, 2772
std::pmr::pool_options, 2773
max_blocks_per_chunk, 2773
std::pmr::unsynchronized_pool_resource, 3214
do_allocate, 3215
do_deallocate, 3215
do_is_equal, 3215
std::pointer_to_binary_function< _Arg1, _Arg2, _Result >, 2762
first_argument_type, 2763
result_type, 2763
second_argument_type, 2763
std::pointer_to_unary_function< _Arg, _Result >, 2763
argument_type, 2764
result_type, 2764
std::pointer_traits< _Ptr >, 2764
difference_type, 2765
element_type, 2765
pointer, 2765
pointer_to, 2765
rebind, 2765
std::pointer_traits< _Tp * >, 2766
difference_type, 2766
element_type, 2766
pointer, 2766, 2767
pointer_to, 2767, 2768
rebind, 2767
std::poisson_distribution< _IntType >, 2768
max, 2770
mean, 2770
min, 2770
operator<<, 2771
operator>>, 2771
operator(), 2770
operator==, 2771
param, 2770
reset, 2771
result_type, 2769
std::poisson_distribution< _IntType >::param_type, 2734
std::predicate, 738
std::priority_queue< _Tp, _Sequence, _Compare >, 2775
empty, 2779
pop, 2779
priority_queue, 2777, 2778
push, 2779
size, 2779
top, 2779
std::promise< _Res >, 2782
std::promise< _Res & >, 2782
std::promise< void >, 2783
std::queue< _Tp, _Sequence >, 2785
back, 2787
c, 2788
empty, 2787
front, 2787
pop, 2788
push, 2788
queue, 2787
size, 2788
std::random_access_iterator_tag, 2790
std::random_device, 2791

result_type, 2792
std::range_error, 2792
what, 2793
std::ranges::bidirectional_range, 739
std::ranges::borrowed_range, 739
std::ranges::common_range, 739
std::ranges::contiguous_range, 739
std::ranges::dangling, 2040
std::ranges::elements_of< _Range, _Alloc >, 2110
std::ranges::empty_view< _Tp >, 2110
std::ranges::equal_to, 2138
std::ranges::forward_range, 739
std::ranges::greater, 2228
std::ranges::greater_equal, 2230
std::ranges::input_range, 740
std::ranges::less, 2345
std::ranges::less_equal, 2347
std::ranges::not_equal_to, 2565
std::ranges::output_range, 740
std::ranges::random_access_range, 740
std::ranges::range, 740
std::ranges::single_view< _Tp >, 2920
std::ranges::sized_range, 741
std::ranges::subrange< _It, _Sent, _Kind >, 2986
std::ranges::view, 741
std::ranges::view_base, 3249
std::ranges::view_interface< _Derived >, 3249
std::ranges::viewable_range, 741
std::rank< _Tp >, 2799
std::ratio< _Num, _Den >, 2800
std::ratio_equal< _R1, _R2 >, 2800
std::ratio_greater< _R1, _R2 >, 2801
std::ratio_greater_equal< _R1, _R2 >, 2802
std::ratio_less< _R1, _R2 >, 2803
std::ratio_less_equal< _R1, _R2 >, 2804
std::ratio_not_equal< _R1, _R2 >, 2804
std::raw_storage_iterator< _OutputIterator, _Tp >, 2805
difference_type, 2806
iterator_category, 2806
pointer, 2807
reference, 2807
value_type, 2807
std::recursive_mutex, 2821
std::recursive_timed_mutex, 2822
std::reference_wrapper< _Tp >, 2824
cref, 2825
ref, 2825
std::regex_constants, 713
__multiline, 719
__polynomial, 719
awk, 720
basic, 720
collate, 720
ECMAScript, 720
egrep, 720
error_backref, 715
error_badbrace, 715
error_badrepeat, 716
error_brace, 716
error_brack, 716
error_collate, 716
error_complexity, 716
error_ctype, 716
error_escape, 716
error_paren, 716
error_range, 716
error_space, 717
error_stack, 717
error_type, 715
extended, 720
format_default, 720
format_first_only, 721
format_no_copy, 721
format_sed, 721
grep, 721
icase, 721
match_any, 721
match_continuous, 721
match_default, 722
match_flag_type, 715
match_not_bol, 722
match_not_bow, 722
match_not_eol, 722
match_not_eow, 722
match_not_null, 722
match_prev_avail, 722
multiline, 722
nosubs, 723
operator&, 717
operator&=, 717
operator~, 719
operator^, 718
operator^=, 718
operator|, 718
operator|=, 719
optimize, 723
syntax_option_type, 715
std::regex_error, 2825
code, 2827
regex_error, 2826
what, 2827
std::regex_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2827
operator++, 2829
operator->, 2829
operator=, 2829
operator==, 2829
operator*, 2829

regex_iterator, 2828
std::regex_token_iterator< _Bi_iter, _Ch_type, _Rx_traits >, 2830
operator++, 2833
operator->, 2833
operator=, 2834
operator==, 2834
operator*, 2833
regex_token_iterator, 2831–2833
std::regex_traits< _Ch_type >, 2834
getloc, 2835
imbue, 2835
isctype, 2836
length, 2836
lookup_classname, 2837
lookup_collatename, 2838
regex_traits, 2835
transform, 2838
transform_primary, 2839
translate, 2839
translate_nocase, 2839
value, 2840
std::regular_invocable, 741
std::rel_ops, 723
operator!=, 723
operator<=, 724
operator>, 724
operator>=, 724
std::relation, 742
std::remove_all_extents< _Tp >, 2840
std::remove_const< _Tp >, 2841
std::remove_cv< _Tp >, 2841
std::remove_extent< _Tp >, 2841
std::remove_pointer< _Tp >, 2841
std::remove_reference< _Tp >, 2842
std::remove_volatile< _Tp >, 2842
std::result_of< _Signature >, 2844
std::reverse_iterator< _Iterator >, 2844
base, 2847
operator+, 2847
operator++, 2847
operator+=, 2848
operator-, 2848
operator->, 2849
operator--, 2848
operator-=, 2848
operator[], 2849
operator*, 2847
reverse_iterator, 2846
std::runtime_error, 2854
runtime_error, 2855
what, 2855
std::same_as, 742
std::scoped_allocator_adaptor< _OuterAlloc, _InnerAllocs >, 2870
std::seed_seq, 2871
result_type, 2872
seed_seq, 2872
std::set< _Key, _Compare, _Alloc >, 2880
~set, 2887
allocator_type, 2883
begin, 2888
cbegin, 2888
cend, 2888
clear, 2888
const_iterator, 2883
const_pointer, 2883
const_reference, 2883
const_reverse_iterator, 2883
contains, 2888
count, 2889
crbegin, 2889
crend, 2889
difference_type, 2883
emplace, 2889
emplace_hint, 2890
empty, 2890
end, 2890
equal_range, 2891
erase, 2891, 2892
find, 2893
get_allocator, 2893
insert, 2894, 2895
iterator, 2883
key_comp, 2896
key_compare, 2883
key_type, 2884
lower_bound, 2896
max_size, 2897
operator=, 2897
pointer, 2884
rbegin, 2897
reference, 2884
rend, 2898
reverse_iterator, 2884
set, 2885–2887
size, 2898
size_type, 2884
swap, 2898
upper_bound, 2898, 2899
value_comp, 2899
value_compare, 2884
value_type, 2884
std::shared_future< _Res >, 2899
_M_get_result, 2901
get, 2901
shared_future, 2900

std::shared_future< _Res & >, 2901
_M_get_result, 2902
get, 2902
shared_future, 2902
std::shared_future< void >, 2903
_M_get_result, 2904
shared_future, 2904
std::shared_lock< _Mutex >, 2904
std::shared_ptr< _Tp >, 2905
element_type, 2908
get, 2914
operator bool, 2914
owner_before, 2914
shared_ptr, 2908–2914
swap, 2914
unique, 2915
use_count, 2915
std::shared_timed_mutex, 2915
std::shuffle_order_engine< _RandomNumberEngine, __k >, 2916
base, 2918
discard, 2918
max, 2918
min, 2918
operator<<, 2919
operator>>, 2920
operator(), 2918
operator==, 2919
result_type, 2917
seed, 2919
shuffle_order_engine, 2917, 2918
std::simd::__vec_builtin, 742
std::simd::__vec_builtin_of, 742
std::simd::rebind< _Tp, _Mp, _Traits >, 2818
std::simd::rebind< _Tp, _Vp, _Traits >, 2819
std::slice, 2922
operator==, 2922
std::slice_array< _Tp >, 2922
operator<<=, 2924
operator>>=, 2924
operator+=, 2924
operator-=, 2924
operator/=, 2924
operator%=, 2924
operator&=, 2924
operator*=, 2924
operator^=, 2925
operator|=, 2925
std::sortable, 743
std::stack< _Tp, _Sequence >, 2933
empty, 2935
pop, 2935
push, 2935
size, 2935
stack, 2935
top, 2935, 2936
std::strict_weak_order, 743
std::student_t_distribution< _RealType >, 2979
max, 2980
min, 2980
operator<<, 2981
operator>>, 2982
operator(), 2980
operator==, 2981
param, 2981
reset, 2981
result_type, 2980
std::student_t_distribution< _RealType >::param_type, 2735
std::sub_match< _BiIter >, 2982
compare, 2984
length, 2985
operator string_type, 2985
str, 2985
swap, 2985
std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2987
discard, 2989
max, 2989
min, 2989
operator<<, 2990
operator>>, 2990
operator(), 2989
operator==, 2990
result_type, 2988
seed, 2989
subtract_with_carry_engine, 2988
std::system_error, 2994
what, 2994
std::this_thread, 725
get_id, 725
sleep_for, 725
sleep_until, 725
yield, 726
std::thread, 3000
native_handle, 3001
std::thread::id, 2270
std::time_base, 3008
std::time_get< _CharT, _InIter >, 3009
~time_get, 3011
char_type, 3011
date_order, 3011
do_date_order, 3012
do_get, 3012
do_get_date, 3013
do_get_monthname, 3013
do_get_time, 3014
do_get_weekday, 3015

do_get_year, 3015
get, 3016, 3017
get_date, 3017
get_monthname, 3018
get_time, 3019
get_weekday, 3019
get_year, 3020
id, 3021
iter_type, 3011
time_get, 3011
std::time_get_byname< _CharT, _InIter >, 3021
date_order, 3023
do_date_order, 3023
do_get, 3023
do_get_date, 3024
do_get_monthname, 3024
do_get_time, 3025
do_get_weekday, 3026
do_get_year, 3026
get, 3027, 3028
get_date, 3029
get_monthname, 3029
get_time, 3030
get_weekday, 3030
get_year, 3031
id, 3032
std::time_put< _CharT, _OutIter >, 3033
~time_put, 3034
char_type, 3034
do_put, 3035
id, 3037
iter_type, 3034
put, 3035, 3036
time_put, 3034
std::time_put_byname< _CharT, _OutIter >, 3037
do_put, 3038
id, 3040
put, 3039
std::timed_mutex, 3040
std::to_chars_result, 3041
std::tr1, 726
arg, 729
conf_hyperg, 729
conf_hypergf, 729
conf_hypergl, 730
hyperg, 730
hypergf, 730
hypergl, 731
norm, 731
polar, 731
std::tr1::__detail, 731
std::tr2, 731
std::tr2::__detail, 733
std::tr2::__dynamic_bitset_base< _WordT, _Alloc >, 780
_M_w, 782
std::tr2::__reflection_typelist< _Elements >, 818
std::tr2::__reflection_typelist< _First, _Rest... >, 818
std::tr2::__reflection_typelist<>, 819
std::tr2::bases< _Tp >, 1110
std::tr2::bool_set, 1813
bool_set, 1814
equals, 1814
is_emptyset, 1814
is_indeterminate, 1814
is_singleton, 1814
operator bool, 1814
std::tr2::direct_bases< _Tp >, 2078
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2096
all, 2101
any, 2101
append, 2101
clear, 2101
count, 2101
dynamic_bitset, 2099–2101
empty, 2102
find_first, 2102
find_next, 2102
flip, 2102
get_allocator, 2103
max_size, 2103
none, 2103
num_blocks, 2103
operator<<, 2104
operator<<=, 2104
operator>>, 2105
operator>>=, 2105
operator-=, 2104
operator=, 2105
operator&=, 2103, 2104
operator[], 2105, 2106
operator~, 2106
operator^=, 2106
operator|=, 2106
push_back, 2107
reset, 2107
resize, 2107
set, 2107
size, 2108
swap, 2108
test, 2108
to_string, 2109
to_ullong, 2109
to_ulong, 2109
std::tr2::dynamic_bitset< _WordT, _Alloc >::reference, 2823
std::try_to_lock_t, 3072
std::tuple< _Elements >, 3072
std::tuple< _T1, _T2 >, 3074

std::tuple_element< 0, pair< _Tp1, _Tp2 > >, 3076
std::tuple_element< 1, pair< _Tp1, _Tp2 > >, 3076
std::tuple_element< __i, _Tp >, 3076
std::tuple_element< __i, tuple< _Types... > >, 3077
std::tuple_element< _Ind, array< _Tp, _Nm > >, 3077
std::tuple_size< _Tp >, 3077
std::tuple_size< array< _Tp, _Nm > >, 3077
std::tuple_size< pair< _Tp1, _Tp2 > >, 3078
std::tuple_size< tuple< _Elements... > >, 3079
std::type_index, 3081
std::type_info, 3082
~type_info, 3083
before, 3083
name, 3083
std::type_order< _Tp, _Up >, 3083
std::unary_function< _Arg, _Result >, 3088
argument_type, 3088
result_type, 3088
std::unary_negate< _Predicate >, 3089
argument_type, 3089
result_type, 3089
std::underflow_error, 3091
what, 3091
std::underlying_type< _Tp >, 3092
std::uniform_int_distribution< _IntType >, 3092
max, 3093
min, 3093
operator(), 3093
operator==, 3094
param, 3094
reset, 3094
result_type, 3093
uniform_int_distribution, 3093
std::uniform_int_distribution< _IntType >::param_type, 2735
std::uniform_random_bit_generator, 743
std::uniform_real_distribution< _RealType >, 3094
max, 3096
min, 3096
operator(), 3096
operator==, 3097
param, 3096
reset, 3097
result_type, 3095
uniform_real_distribution, 3096
std::uniform_real_distribution< _RealType >::param_type, 2736
std::unique_lock< _Mutex >, 3097
swap, 3098
std::unique_ptr< _Tp, _Dp >, 3098
~unique_ptr, 3101
get, 3102
get_deleter, 3102
operator bool, 3102
operator->, 3102
operator=, 3102, 3103
operator*, 3102
release, 3103
reset, 3103
swap, 3103
unique_ptr, 3100, 3101
std::unique_ptr< _Tp[], _Dp >, 3104
~unique_ptr, 3107
get, 3107
get_deleter, 3107
operator bool, 3107
operator<<, 3109
operator->, 3108
operator=, 3108
operator[], 3108
operator*, 3107
release, 3108
reset, 3109
swap, 3109
unique_ptr, 3105–3107
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3112
allocator_type, 3116
at, 3120–3122
begin, 3122, 3123
bucket_count, 3123
cbegin, 3123
cend, 3124
clear, 3124
const_iterator, 3116
const_local_iterator, 3116
const_pointer, 3116
const_reference, 3117
contains, 3124, 3125
count, 3125
difference_type, 3117
emplace, 3125
emplace_hint, 3126
empty, 3126
end, 3127
equal_range, 3127, 3128
erase, 3128–3130
find, 3130, 3131
get_allocator, 3131
hash_function, 3131
hasher, 3117
insert, 3131–3134
iterator, 3117
key_eq, 3135
key_equal, 3117
key_type, 3117
load_factor, 3135
local_iterator, 3117

mapped_type, 3117
max_bucket_count, 3135
max_load_factor, 3135
max_size, 3136
operator=, 3136
operator[], 3136, 3137
pointer, 3118
reference, 3118
rehash, 3138
reserve, 3138
size, 3138
size_type, 3118
swap, 3138
unordered_map, 3118–3120
value_type, 3118
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3141
allocator_type, 3145
begin, 3150
bucket_count, 3150
cbegin, 3151
cend, 3151
clear, 3152
const_iterator, 3145
const_local_iterator, 3145
const_pointer, 3145
const_reference, 3146
contains, 3152
count, 3152
difference_type, 3146
emplace, 3153
emplace_hint, 3153
empty, 3154
end, 3154
equal_range, 3155
erase, 3155–3157
find, 3157, 3158
get_allocator, 3158
hash_function, 3158
hasher, 3146
insert, 3158–3161
iterator, 3146
key_eq, 3162
key_equal, 3146
key_type, 3146
load_factor, 3162
local_iterator, 3146
mapped_type, 3146
max_bucket_count, 3162
max_load_factor, 3162
max_size, 3163
operator=, 3163
pointer, 3147
reference, 3147
rehash, 3163
reserve, 3164
size, 3164
size_type, 3147
swap, 3164
unordered_multimap, 3147–3149
value_type, 3147
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3167
allocator_type, 3170
begin, 3175
bucket_count, 3176
cbegin, 3176
cend, 3176
clear, 3177
const_iterator, 3170
const_local_iterator, 3171
const_pointer, 3171
const_reference, 3171
contains, 3177
count, 3178
difference_type, 3171
emplace, 3178
emplace_hint, 3178
empty, 3179
end, 3179, 3180
equal_range, 3180
erase, 3181, 3182
find, 3182, 3183
get_allocator, 3183
hash_function, 3183
hasher, 3171
insert, 3183–3185
iterator, 3171
key_eq, 3186
key_equal, 3171
key_type, 3172
load_factor, 3186
local_iterator, 3172
max_bucket_count, 3186
max_load_factor, 3186
max_size, 3186
operator=, 3187
pointer, 3172
reference, 3172
rehash, 3187
reserve, 3188
size, 3188
size_type, 3172
swap, 3188
unordered_multiset, 3172–3174
value_type, 3172
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3191

allocator_type, 3194
begin, 3199
bucket_count, 3200
cbegin, 3200
cend, 3200
clear, 3201
const_iterator, 3194
const_local_iterator, 3195
const_pointer, 3195
const_reference, 3195
contains, 3201
count, 3202
difference_type, 3195
emplace, 3202
emplace_hint, 3202
empty, 3203
end, 3203, 3204
equal_range, 3204
erase, 3205, 3206
find, 3206, 3207
get_allocator, 3207
hash_function, 3207
hasher, 3195
insert, 3207–3210
iterator, 3195
key_eq, 3211
key_equal, 3195
key_type, 3196
load_factor, 3211
local_iterator, 3196
max_bucket_count, 3211
max_load_factor, 3211
max_size, 3212
operator=, 3212
pointer, 3196
reference, 3196
rehash, 3212
reserve, 3213
size, 3213
size_type, 3196
swap, 3213
unordered_set, 3196–3198
value_type, 3196
std::uses_allocator< tuple< _Types... >, _Alloc >, 3215
std::uses_allocator< typename, typename >, 3215
std::valarray< _Tp >, 3216
begin, 3219
end, 3219
operator!, 3219
operator<<=, 3221
operator>>=, 3221
operator+, 3220
operator+=, 3220
operator-, 3220
operator-=, 3220, 3221
operator/=, 3221
operator%=, 3219
operator&=, 3219
operator*=, 3220
operator~, 3222
operator^=, 3222
operator|=, 3222
valarray, 3219
std::vector< _Tp, _Alloc >, 3225
_M_allocate_and_copy, 3232
_M_range_check, 3232
~vector, 3232
assign, 3232, 3233
at, 3233, 3234
back, 3234
begin, 3234
capacity, 3235
cbegin, 3235
cend, 3235
clear, 3235
crbegin, 3235
crend, 3235
data, 3236
emplace, 3236
empty, 3236
end, 3236
erase, 3237
front, 3237, 3238
get_allocator, 3238
insert, 3238–3240
max_size, 3240
operator=, 3240, 3241
operator[], 3241
pop_back, 3242
push_back, 3242
rbegin, 3242
rend, 3243
reserve, 3243
resize, 3243, 3244
shrink_to_fit, 3244
size, 3244
swap, 3244
vector, 3229–3231
std::vector< bool, _Alloc >, 3245
_M_allocate_and_copy, 3249
get_allocator, 3249
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3250
_M_buf_locale, 3266
_M_in_beg, 3267
_M_in_cur, 3267
_M_in_end, 3267
_M_out_beg, 3267
_M_out_cur, 3267

_M_out_end, 3267
__streambuf_type, 3252
char_type, 3252
eback, 3254
egptr, 3254
epptr, 3254
gbump, 3254
getloc, 3255
gptr, 3255
imbue, 3255
in_avail, 3256
int_type, 3252
off_type, 3253
overflow, 3256
pbackfail, 3257
pbase, 3257
pbump, 3257
pos_type, 3253
pptr, 3258
pubimbue, 3258
pubseekoff, 3258
pubseekpos, 3259
pubsetbuf, 3259
pubsync, 3259
sbumpc, 3259
seekoff, 3260
seekpos, 3260
setbuf, 3260
setg, 3261
setp, 3261
sgetc, 3261
sgetn, 3262
showmanyc, 3262
snextc, 3263
sputbackc, 3263
sputc, 3263
sputn, 3264
state, 3264
sungetc, 3264
sync, 3264
traits_type, 3253
uflow, 3265
underflow, 3265
wbuffer_convert, 3253
xsgetn, 3265
xsputn, 3266
std::weak_ptr< _Tp >, 3267
std::weakly_incrementable, 744
std::weibull_distribution< _RealType >, 3268
a, 3270
b, 3270
max, 3270
min, 3270
operator(), 3270
operator==, 3271
param, 3270
reset, 3271
result_type, 3270
std::weibull_distribution< _RealType >::param_type, 2736
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3271
converted, 3273
from_bytes, 3273
state, 3274
to_bytes, 3274
wstring_convert, 3272
std_abs.h, 3396
std_function.h, 3397
std_mutex.h, 3397
std_thread.h, 3398
stdatomic.h, 3684
stdbit.h, 3684
stdc++.h, 3725
stdckdint.h, 3685
stddev
std::normal_distribution< _RealType >, 2562
stdexcept, 3685
stdexcept_except.h, 3398
stdexcept_throw.h, 3399
stdexcept_throwdef.h, 3399
stdexcept_throwfwd.h, 3399
stdio_filebuf
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2939
stdio_filebuf.h, 3599
stdio_sync_filebuf.h, 3599
stdlib.h, 3685
abs, 3685
stdtr1c++.h, 3725
stl_algo.h, 3399
stl_algobase.h, 3406
stl_bvector.h, 3411
stl_construct.h, 3412
stl_deque.h, 3413
_GLIBCXX_DEQUE_BUF_SIZE, 3413
stl_function.h, 3414
stl_heap.h, 3416
stl_iterator.h, 3417, 3420
stl_iterator_base_funcs.h, 3421
stl_iterator_base_types.h, 3422
stl_list.h, 3422
stl_map.h, 3423
stl_multimap.h, 3424
stl_multiset.h, 3425
stl_numeric.h, 3426
stl_pair.h, 3427
stl_queue.h, 3428
stl_raw_storage_iter.h, 3429

stl_relops.h, 3429
stl_set.h, 3430
stl_stack.h, 3430
stl_tempbuf.h, 3431
stl_tree.h, 3432
stl_uninitialized.h, 3432
stl_vector.h, 3433
stop_token, 3685
str
    std::basic_istringstream< _CharT, _Traits, _Alloc >, 1369
    std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1468
    std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1708
    std::basic_stringstream< _CharT, _Traits, _Alloc >, 1751
    std::match_results< _Bi_iter, _Alloc >, 2445
    std::sub_match< _BiIter >, 2985
stream_iterator.h, 3434
streambuf, 3685
    I/O, 170
streambuf.tcc, 3434
streambuf_iterator.h, 3435
streambuf_type
    std::istreambuf_iterator< _CharT, _Traits >, 2326
    std::ostreambuf_iterator< _CharT, _Traits >, 2694
streamoff
    std, 588
streampos
    std, 588
streamsize
    std, 588
stride
    Numeric Arrays, 236
string, 3686, 3688, 3689
    Strings, 278
string_conversions.h, 3599
string_type
    std::__cxx11::collate< _CharT >, 1895
    std::__cxx11::collate_byname< _CharT >, 1900
    std::messages< _CharT >, 2460
    std::money_get< _CharT, _InIter >, 2472
    std::money_put< _CharT, _OutIter >, 2476
    std::moneypunct< _CharT, _Intl >, 2482
    std::numpunct< _CharT >, 2677
string_view, 3689, 3691
string_view.tcc, 3435, 3436
stringbuf
    I/O, 170
stringfwd.h, 3436
Strings, 278
    string, 278
    u16string, 278
    u32string, 279
    wstring, 279
stringstream
    I/O, 170
strstream, 3285
substr
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 879
    __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1579
    std::basic_string< _CharT, _Traits, _Alloc >, 1684
subtract_with_carry_engine
    std::subtract_with_carry_engine< _UIntType, __w, __s, __r >, 2988
subtractive_rng
    __gnu_cxx::subtractive_rng, 2992
subview
    __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1580
    std::basic_string< _CharT, _Traits, _Alloc >, 1685
suffix
    std::match_results< _Bi_iter, _Alloc >, 2445
sum
    Numeric Arrays, 236
sungetc
    __gnu_cxx::enc_filebuf< _CharT >, 2125
    __gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2953
    __gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2971
    std::basic_filebuf< _CharT, _Traits >, 1128
    std::basic_streambuf< _CharT, _Traits >, 1501
    std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1708
    std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3264
swap
    __gnu_cxx, 413
    __gnu_cxx::__versa_string< _CharT, _Traits, _Alloc, _Base >, 879
    __gnu_cxx::pair< _T1, _T2 >, 2716
    __gnu_debug::basic_string< _CharT, _Traits, _Allocator >, 1580
    __gnu_parallel::_IteratorPair< _Iterator1, _Iterator2, _IteratorCategory >, 919, 920
    __gnu_pbds::sample_probe_fn, 2856
    __gnu_pbds::sample_range_hashing, 2857
    __gnu_pbds::sample_ranged_hash_fn, 2858
    __gnu_pbds::sample_resize_policy, 2861
    __gnu_pbds::sample_resize_trigger, 2864
    __gnu_pbds::sample_size_policy, 2865
    __gnu_pbds::sample_update_policy, 2869
    Futures, 121
    Metaprogramming, 361
    Numeric Arrays, 236
    Pointer Abstractions, 352
    Regular Expressions, 276
    std, 645–648

std::__debug, 657
std::basic_regex< _Ch_type, _Rx_traits >, 1486
std::basic_string< _CharT, _Traits, _Alloc >, 1686
std::copyable_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 1942
std::deque< _Tp, _Alloc >, 2077
std::experimental::fundamentals_v1::any, 1078
std::forward_list< _Tp, _Alloc >, 2187
std::function< _Res(_ArgTypes...)>, 2199
std::list< _Tp, _Alloc >, 2378
std::map< _Key, _Tp, _Compare, _Alloc >, 2432
std::match_results< _Bi_iter, _Alloc >, 2446
std::move_only_function< _Res(_ArgTypes...) _GLIBCXX_MOF_CV noexcept(_Noex)>, 2502, 2503
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2525
std::multiset< _Key, _Compare, _Alloc >, 2546
std::pair< _T1, _T2 >, 2721
std::set< _Key, _Compare, _Alloc >, 2898
std::shared_ptr< _Tp >, 2914
std::sub_match< _BiIter >, 2985
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2108
std::unique_lock< _Mutex >, 3098
std::unique_ptr< _Tp, _Dp >, 3103
std::unique_ptr< _Tp[], _Dp >, 3109
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3138
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3164
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3188
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3213
std::vector< _Tp, _Alloc >, 3244
Type-safe container of any type, 297
Utilities, 312, 313
swap_ranges
Mutating, 39
sync
__gnu_cxx::enc_filebuf< _CharT >, 2125
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2954
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2971
std::basic_filebuf< _CharT, _Traits >, 1128
std::basic_fstream< _CharT, _Traits >, 1174
std::basic_ifstream< _CharT, _Traits >, 1218
std::basic_iostream< _CharT, _Traits >, 1289
std::basic_istream< _CharT, _Traits >, 1329
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1370
std::basic_streambuf< _CharT, _Traits >, 1501
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1708
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1751
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3264
sync_with_stdio
std::basic_fstream< _CharT, _Traits >, 1174
std::basic_ifstream< _CharT, _Traits >, 1219
std::basic_ios< _CharT, _Traits >, 1244
std::basic_iostream< _CharT, _Traits >, 1289
std::basic_istream< _CharT, _Traits >, 1329
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1370
std::basic_ofstream< _CharT, _Traits >, 1403
std::basic_ostream< _CharT, _Traits >, 1435
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1468
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1752
std::ios_base, 2295
syncstream, 3692
syntax_option_type
std::regex_constants, 715
synth_access_traits
__gnu_pbds::detail::trie_traits< Key, Mapped, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3070
__gnu_pbds::detail::trie_traits< Key, null_type, _ATraits, Node_Update, pat_trie_tag, _Alloc >, 3072
synth_access_traits.hpp, 3573
system_category
Diagnostics, 132
system_error, 3692, 3693

t
std::binomial_distribution< _IntType >, 1793
table
std::ctype< char >, 1980
std::ctype_byname< char >, 2035
table_size
std::ctype< char >, 1988
std::ctype_byname< char >, 2039
tag_and_trait.hpp, 3583
Tags, 149
trivial_iterator_difference_type, 149
tags.h, 3673
tan
Complex Numbers, 190
math.h, 3626
tanh
Complex Numbers, 190
math.h, 3627
target
std::function< _Res(_ArgTypes...)>, 2199, 2200
target_type
std::function< _Res(_ArgTypes...)>, 2200

Technical Specifications, 279
tellg
std::basic_fstream< _CharT, _Traits >, 1175
std::basic_ifstream< _CharT, _Traits >, 1219
std::basic_iostream< _CharT, _Traits >, 1289
std::basic_istream< _CharT, _Traits >, 1330
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1370
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1752
tellp
std::basic_fstream< _CharT, _Traits >, 1175
std::basic_iostream< _CharT, _Traits >, 1290
std::basic_ofstream< _CharT, _Traits >, 1404
std::basic_ostream< _CharT, _Traits >, 1435
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1469
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1752
temporary_buffer
__gnu_cxx::temporary_buffer< _ForwardIterator, _Tp >, 2996
terminate
Exceptions, 136
terminate_handler
Exceptions, 134
test
std::bitset< _Nb >, 1812
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2108
text_encoding, 3693
text_encoding-data.h, 3436
tgmath.h, 3693
thin_heap_.hpp, 3577
this_thread_sleep.h, 3436
thousands_sep
std::moneypunct< _CharT, _Intl >, 2488
std::moneypunct_byname< _CharT, _Intl >, 2496
std::numpunct< _CharT >, 2680
std::numpunct_byname< _CharT >, 2684
thread, 3694
Threads, 124
throw_allocator.h, 3600
throw_with_nested
Exceptions, 136
tie
std::basic_fstream< _CharT, _Traits >, 1175, 1176
std::basic_ifstream< _CharT, _Traits >, 1219
std::basic_ios< _CharT, _Traits >, 1244
std::basic_iostream< _CharT, _Traits >, 1290
std::basic_istream< _CharT, _Traits >, 1330
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1371
std::basic_ofstream< _CharT, _Traits >, 1404
std::basic_ostream< _CharT, _Traits >, 1436
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1469
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1753
Utilities, 313
Time, 365
abs, 375
ceil, 375, 376
clock_cast, 376
days, 374
duration_cast, 377
floor, 377, 378
high_resolution_clock, 374
hours, 374
is_am, 378
is_pm, 378
local_time_format, 379
make12, 379
make24, 379
microseconds, 374
milliseconds, 374
minutes, 374
months, 374
nanoseconds, 375
operator<, 383
operator<<, 383
operator<=, 383
operator<=>, 384
operator>, 384
operator>=, 384
operator+, 381, 382
operator-, 382
operator/, 382, 383
operator==, 384
operator%, 381
operator*, 381
round, 384, 385
seconds, 375
time_point_cast, 385
weeks, 375
years, 375
time
std::locale, 2389
time_get
std::time_get< _CharT, _InIter >, 3011
time_members.h, 3725
time_point_cast
Time, 385
time_put
std::time_put< _CharT, _OutIter >, 3034
tinyness_before
std::__numeric_limits_base, 809
std::numeric_limits< _Tp >, 2603
std::numeric_limits< bool >, 2607

std::numeric_limits< char >, 2611
std::numeric_limits< char16_t >, 2615
std::numeric_limits< char32_t >, 2619
std::numeric_limits< double >, 2623
std::numeric_limits< float >, 2627
std::numeric_limits< int >, 2631
std::numeric_limits< long >, 2635
std::numeric_limits< long double >, 2639
std::numeric_limits< long long >, 2643
std::numeric_limits< short >, 2647
std::numeric_limits< signed char >, 2651
std::numeric_limits< unsigned char >, 2655
std::numeric_limits< unsigned int >, 2659
std::numeric_limits< unsigned long >, 2663
std::numeric_limits< unsigned long long >, 2667
std::numeric_limits< unsigned short >, 2671
std::numeric_limits< wchar_t >, 2675
TLB_size
__gnu_parallel::_Settings, 1036
to_address
Pointer Abstractions, 352
to_array
Array creation functions, 287
to_bytes
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3274
to_string
std::bitset< _Nb >, 1812
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2109
to_ullong
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2109
to_ulong
std::bitset< _Nb >, 1812
std::tr2::dynamic_bitset< _WordT, _Alloc >, 2109
Todo List, 2
tolower
std, 648
std::__ctype_abstract_base< _CharT >, 777, 778
std::ctype< _CharT >, 1954, 1955
std::ctype< char >, 1980–1982
std::ctype< wchar_t >, 1999
std::ctype_byname< _CharT >, 2013, 2014
std::ctype_byname< char >, 2035, 2036
top
std::priority_queue< _Tp, _Sequence, _Compare >, 2779
std::stack< _Tp, _Sequence >, 2935, 2936
toupper
std, 648
std::__ctype_abstract_base< _CharT >, 778
std::ctype< _CharT >, 1955
std::ctype< char >, 1983–1985
std::ctype< wchar_t >, 1999, 2000
std::ctype_byname< _CharT >, 2014
std::ctype_byname< char >, 2036–2038
TR1 Mathematical Special Functions, 250
assoc_laguerre, 252
assoc_legendre, 252
beta, 252
comp_ellint_1, 252
comp_ellint_2, 253
comp_ellint_3, 253
cyl_bessel_i, 253
cyl_bessel_j, 253
cyl_bessel_k, 253
cyl_neumann, 253
ellint_1, 253
ellint_2, 254
ellint_3, 254
expint, 254
hermite, 254
laguerre, 254
legendre, 254
riemann_zeta, 254
sph_bessel, 255
sph_legendre, 255
sph_neumann, 255
trace_fn_imps.hpp, 3555, 3556
Traits, 151
traits.hpp, 3541, 3542
traits_type
std::basic_ios< _CharT, _Traits >, 1234
std::basic_istream< _CharT, _Traits >::sentry, 2875
std::basic_streambuf< _CharT, _Traits >, 1490
std::istreambuf_iterator< _CharT, _Traits >, 2326
std::ostream_iterator< _Tp, _CharT, _Traits >, 2691
std::ostreambuf_iterator< _CharT, _Traits >, 2695
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3253
transform
Mutating, 39, 40
std::__cxx11::collate< _CharT >, 1898
std::regex_traits< _Ch_type >, 2838
transform_exclusive_scan
Generalized Numeric operations, 16
transform_inclusive_scan
Generalized Numeric operations, 17, 18
transform_minimal_n
__gnu_parallel::_Settings, 1036
transform_primary
std::regex_traits< _Ch_type >, 2839
transform_reduce
Generalized Numeric operations, 19, 20
translate
std::regex_traits< _Ch_type >, 2839
translate_nocase
std::regex_traits< _Ch_type >, 2839
traps
std::__numeric_limits_base, 809

std::numeric_limits< _Tp >, 2603
std::numeric_limits< bool >, 2607
std::numeric_limits< char >, 2611
std::numeric_limits< char16_t >, 2615
std::numeric_limits< char32_t >, 2619
std::numeric_limits< double >, 2623
std::numeric_limits< float >, 2627
std::numeric_limits< int >, 2631
std::numeric_limits< long >, 2635
std::numeric_limits< long double >, 2639
std::numeric_limits< long long >, 2643
std::numeric_limits< short >, 2647
std::numeric_limits< signed char >, 2651
std::numeric_limits< unsigned char >, 2655
std::numeric_limits< unsigned int >, 2659
std::numeric_limits< unsigned long >, 2663
std::numeric_limits< unsigned long long >, 2667
std::numeric_limits< unsigned short >, 2671
std::numeric_limits< wchar_t >, 2675
tree
__gnu_pbds::tree< Key, Mapped, Cmp_Fn, Tag, Node_Update, _Alloc >, 3044
tree_policy.hpp, 3584
tree_trace_base.hpp, 3579
trie
__gnu_pbds::trie< Key, Mapped, _ATraits, Tag, Node_Update, _Alloc >, 3057
trie_policy.hpp, 3585
trie_policy_base.hpp, 3580
trie_string_access_traits_imp.hpp, 3580
trivial_iterator_difference_type
Tags, 149
true_type
Metaprogramming, 360
truename
std::numpunct< _CharT >, 2680
std::numpunct_byname< _CharT >, 2684
trunc
std::basic_fstream< _CharT, _Traits >, 1183
std::basic_ifstream< _CharT, _Traits >, 1226
std::basic_ios< _CharT, _Traits >, 1251
std::basic_iostream< _CharT, _Traits >, 1298
std::basic_istream< _CharT, _Traits >, 1337
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1378
std::basic_ofstream< _CharT, _Traits >, 1411
std::basic_ostream< _CharT, _Traits >, 1443
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1476
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1760
std::ios_base, 2301
try_lock
Mutexes, 123
try_to_lock
Mutexes, 124
tuple, 3696, 3697
tuple_cat
Utilities, 313
type
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, cc_hash_tag, Policy_Tl >, 1924
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, gp_hash_tag, Policy_Tl >, 1925
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, list_update_tag, Policy_Tl >, 1925
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, ov_tree_tag, Policy_Tl >, 1926
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, rb_tree_tag, Policy_Tl >, 1926
__gnu_pbds::detail::container_base_dispatch< Key, Mapped, _Alloc, splay_tree_tag, Policy_Tl >, 1927
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, cc_hash_tag, Policy_Tl >, 1927
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, gp_hash_tag, Policy_Tl >, 1928
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, list_update_tag, Policy_Tl >, 1928
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, ov_tree_tag, Policy_Tl >, 1929
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, pat_trie_tag, Policy_Tl >, 1929
__gnu_pbds::detail::container_base_dispatch< Key, null_type, _Alloc, splay_tree_tag, Policy_Tl >, 1930
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binary_heap_tag, null_type >, 1922
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, binomial_heap_tag, null_type >, 1922
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, pairing_heap_tag, null_type >, 1923
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, rc_binomial_heap_tag, null_type >, 1923
__gnu_pbds::detail::container_base_dispatch< _VTp, Cmp_Fn, _Alloc, thin_heap_tag,

null_type >, 1924
__gnu_pbds::detail::default_comb_hash_fn, 2046
__gnu_pbds::detail::default_eq_fn< Key >, 2048
__gnu_pbds::detail::default_hash_fn< Key >, 2049
__gnu_pbds::detail::default_probe_fn< Comb_Probe_Fn >, 2050
__gnu_pbds::detail::default_resize_policy< Comb_Hash_Fn >, 2051
__gnu_pbds::detail::default_trie_access_traits< std::basic_string< Char, Char_Traits, std::allocator< char > > >, 2052
__gnu_pbds::detail::default_update_policy, 2052
__gnu_pbds::detail::entry_cmp< _VTp, Cmp_Fn, _Alloc, true >, 2134
Metaprogramming, 360
std::experimental::fundamentals_v1::any, 1078
Type-safe container of any type, 293
any_cast, 294–296
swap, 297
type_order_v
Variable Templates, 363
type_traits, 3698, 3702
type_traits.h, 3601
type_utils.hpp, 3580
typeindex, 3705
typeinfo, 3730
typelist.h, 3601
types.h, 3673
types_traits.hpp, 3580

u16streampos
std, 588
u16string
__gnu_debug, 421
Strings, 278
u32streampos
std, 588
u32string
__gnu_debug, 421
Strings, 279
u8path
Filesystem, 167
uflow
__gnu_cxx::enc_filebuf< _CharT >, 2126
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2954
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2972
std::basic_filebuf< _CharT, _Traits >, 1129
std::basic_streambuf< _CharT, _Traits >, 1501
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1709
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3265
uncaught_exception
Exceptions, 137
underflow
__gnu_cxx::enc_filebuf< _CharT >, 2126
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2954
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2972
std::basic_filebuf< _CharT, _Traits >, 1129
std::basic_streambuf< _CharT, _Traits >, 1501
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1709
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3265
underlying_type_t
Metaprogramming, 361
unget
std::basic_fstream< _CharT, _Traits >, 1176
std::basic_ifstream< _CharT, _Traits >, 1220
std::basic_iostream< _CharT, _Traits >, 1291
std::basic_istream< _CharT, _Traits >, 1330
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1371
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1753
unicode-data.h, 3437
unicode.h, 3437
Uniform Distributions, 245
operator<<, 245, 246
operator>>, 246, 247
uniform_int_dist.h, 3439
uniform_int_distribution
std::uniform_int_distribution< _IntType >, 3093
uniform_real_distribution
std::uniform_real_distribution< _RealType >, 3096
uninitialized_copy
Memory, 327
uninitialized_copy_n
Memory, 328
SGI, 161
uninitialized_default_construct
Memory, 328
uninitialized_default_construct_n
Memory, 329
uninitialized_fill
Memory, 329
uninitialized_fill_n
Memory, 329
uninitialized_move
Memory, 330
uninitialized_move_n
Memory, 330
uninitialized_value_construct
Memory, 331
uninitialized_value_construct_n
Memory, 331
unique
Mutating, 40, 41
std::forward_list< _Tp, _Alloc >, 2187, 2188
std::list< _Tp, _Alloc >, 2378

std::shared_ptr< _Tp >, 2915
unique_copy
Mutating, 41, 42
unique_copy.h, 3674
unique_copy_minimal_n
__gnu_parallel::_Settings, 1036
unique_lock.h, 3439
unique_ptr
std::unique_ptr< _Tp, _Dp >, 3100, 3101
std::unique_ptr< _Tp[], _Dp >, 3105–3107
unique_ptr.h, 3439
unitbuf
std, 648
std::basic_fstream< _CharT, _Traits >, 1183
std::basic_ifstream< _CharT, _Traits >, 1226
std::basic_ios< _CharT, _Traits >, 1251
std::basic_iostream< _CharT, _Traits >, 1298
std::basic_istream< _CharT, _Traits >, 1337
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1378
std::basic_ofstream< _CharT, _Traits >, 1411
std::basic_ostream< _CharT, _Traits >, 1443
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1476
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1760
std::ios_base, 2301
Unordered Associative, 128
unordered_map, 3706, 3708
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3118–3120
unordered_map.h, 3440
unordered_multimap
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3147–3149
unordered_multiset
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3172–3174
unordered_set, 3709, 3711
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3196–3198
unordered_set.h, 3443
unsetf
std::basic_fstream< _CharT, _Traits >, 1176
std::basic_ifstream< _CharT, _Traits >, 1220
std::basic_ios< _CharT, _Traits >, 1245
std::basic_iostream< _CharT, _Traits >, 1291
std::basic_istream< _CharT, _Traits >, 1331
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1372
std::basic_ofstream< _CharT, _Traits >, 1405
std::basic_ostream< _CharT, _Traits >, 1436
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1469
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1753
std::ios_base, 2296
unshift
std::__codecvt_abstract_base< _InternT, _ExternT, _StateT >, 762
std::codecvt< _InternT, _ExternT, _StateT >, 1849
std::codecvt< _InternT, _ExternT, encoding_state >, 1856, 1857
std::codecvt< char, char, mbstate_t >, 1864
std::codecvt< char16_t, char, mbstate_t >, 1871, 1872
std::codecvt< char32_t, char, mbstate_t >, 1879
std::codecvt< wchar_t, char, mbstate_t >, 1886, 1887
std::codecvt_byname< _InternT, _ExternT, _StateT >, 1893
update_fn_imps.hpp, 3573
upper_bound
Binary Search, 88, 89
std::map< _Key, _Tp, _Compare, _Alloc >, 2432, 2433
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2525
std::multiset< _Key, _Compare, _Alloc >, 2546, 2547
std::set< _Key, _Compare, _Alloc >, 2898, 2899
uppercase
std, 648
std::basic_fstream< _CharT, _Traits >, 1183
std::basic_ifstream< _CharT, _Traits >, 1226
std::basic_ios< _CharT, _Traits >, 1251
std::basic_iostream< _CharT, _Traits >, 1298
std::basic_istream< _CharT, _Traits >, 1337
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1378
std::basic_ofstream< _CharT, _Traits >, 1411
std::basic_ostream< _CharT, _Traits >, 1443
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1476
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1760
std::ios_base, 2301
use_count
std::shared_ptr< _Tp >, 2915
use_facet
Locales, 178
std::locale, 2388
std::locale::id, 2269
uses_allocator_args.h, 3445
Utilities, 301
__addressof, 305
__invoke, 305
addressof, 305
declval, 306

forward, 306
forward_as_tuple, 307
forward_like, 308
get, 308
make_pair, 309
make_tuple, 309
move, 309
move_if_noexcept, 311
operator<=>, 311
operator==, 311, 312
pair, 304
piecewise_construct, 313
swap, 312, 313
tie, 313
tuple_cat, 313
utility, 3711, 3712
utility.h, 3445

valarray, 3712
Numeric Arrays, 224, 225
std::valarray< _Tp >, 3219
valarray_after.h, 3446
valarray_array.h, 3456
valarray_array.tcc, 3464
valarray_before.h, 3465
valid_prefix
__gnu_pbds::detail::pat_trie_base::_Node_citer< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 958
__gnu_pbds::detail::pat_trie_base::_Node_iter< Node, Leaf, Head, Inode, _CIterator, Iterator, _Alloc >, 961
value
std::error_code, 2142
std::error_condition, 2144
std::regex_traits< _Ch_type >, 2840
value_comp
std::map< _Key, _Tp, _Compare, _Alloc >, 2433
std::multimap< _Key, _Tp, _Compare, _Alloc >, 2526
std::multiset< _Key, _Compare, _Alloc >, 2547
std::set< _Key, _Compare, _Alloc >, 2899
value_compare
std::set< _Key, _Compare, _Alloc >, 2884
value_type
__gnu_pbds::detail::bin_search_tree_const_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1767
__gnu_pbds::detail::bin_search_tree_node_it_< Node, Const_Iterator, Iterator, _Alloc >, 1772
__gnu_pbds::detail::binary_heap_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1782
__gnu_pbds::detail::binary_heap_point_const_iterator_< Value_Type, Entry, Simple, _Alloc >, 1785
__gnu_pbds::detail::left_child_next_sibling_heap_const_iterator_< Node, _Alloc >, 2337
__gnu_pbds::detail::left_child_next_sibling_heap_node_point_const_ite
Node, _Alloc >, 2341
std::allocator_traits< _Alloc >, 1057
std::allocator_traits< allocator< _Tp > >, 1062
std::allocator_traits< allocator< void > >, 1067
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1071
std::back_insert_iterator< _Container >, 1099
std::complex< _Tp >, 1904
std::front_insert_iterator< _Container >, 2193
std::insert_iterator< _Container >, 2283
std::istream_iterator< _Tp, _CharT, _Traits, _Dist >, 2323
std::istreambuf_iterator< _CharT, _Traits >, 2326
std::iterator< _Category, _Tp, _Distance, _Pointer, _Reference >, 2332
std::ostream_iterator< _Tp, _CharT, _Traits >, 2691
std::ostreambuf_iterator< _CharT, _Traits >, 2695
std::raw_storage_iterator< _OutputIterator, _Tp >, 2807
std::set< _Key, _Compare, _Alloc >, 2884
std::unordered_map< _Key, _Tp, _Hash, _Pred, _Alloc >, 3118
std::unordered_multimap< _Key, _Tp, _Hash, _Pred, _Alloc >, 3147
std::unordered_multiset< _Value, _Hash, _Pred, _Alloc >, 3172
std::unordered_set< _Value, _Hash, _Pred, _Alloc >, 3196
Variable template for type traits, 297
Variable Templates, 362
is_layout_compatible_v, 363
is_pointer_interconvertible_base_of_v, 363
type_order_v, 363
variant, 3712
vector, 3712–3714
std::__debug::vector< _Tp, _Allocator >, 3225
std::vector< _Tp, _Alloc >, 3229–3231
vector.tcc, 3465
version.h, 3465
void_pointer
__gnu_cxx::__alloc_traits< _Alloc, typename >, 751
std::allocator_traits< _Alloc >, 1057
std::allocator_traits< allocator< _Tp > >, 1062
std::allocator_traits< allocator< void > >, 1067
std::allocator_traits< pmr::polymorphic_allocator< _Tp > >, 1071
void_t
Detection idiom, 291
vstring.h, 3602
vstring.tcc, 3605
vstring_fwd.h, 3606

vstring_util.h, 3606

wbuffer_convert
    std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3253
wcerr
    std, 650
wcin
    std, 650
wclog
    std, 650
wcout
    std, 650
wcregex_token_iterator
    Regular Expressions, 259
wcsub_match
    Regular Expressions, 259
weeks
    Time, 375
wfilebuf
    I/O, 170
wfstream
    I/O, 170
what
    __gnu_cxx::forced_error, 2166
    __gnu_cxx::recursive_init_error, 2821
    __gnu_pbds::container_error, 1931
    __gnu_pbds::insert_error, 2281
    __gnu_pbds::join_error, 2334
    __gnu_pbds::resize_error, 2843
    std::bad_alloc, 1101
    std::bad_cast, 1102
    std::bad_exception, 1103
    std::bad_function_call, 1104
    std::bad_typeid, 1106
    std::bad_weak_ptr, 1107
    std::domain_error, 2094
    std::exception, 2147
    std::experimental::filesystem::v1::filesystem_error, 2160
    std::experimental::fundamentals_v1::bad_any_cast, 1102
    std::experimental::fundamentals_v1::bad_optional_access, 1105
    std::filesystem::filesystem_error, 2162
    std::future_error, 2208
    std::invalid_argument, 2286
    std::ios_base::failure, 2158
    std::length_error, 2343
    std::logic_error, 2391
    std::out_of_range, 2697
    std::overflow_error, 2706
    std::range_error, 2793
    std::regex_error, 2827
    std::runtime_error, 2855
    std::system_error, 2994
    std::underflow_error, 3091
widen
    std::__ctype_abstract_base< _CharT >, 779
    std::basic_fstream< _CharT, _Traits >, 1177
    std::basic_ifstream< _CharT, _Traits >, 1220
    std::basic_ios< _CharT, _Traits >, 1245
    std::basic_iostream< _CharT, _Traits >, 1291
    std::basic_istream< _CharT, _Traits >, 1331
    std::basic_istringstream< _CharT, _Traits, _Alloc >, 1372
    std::basic_ofstream< _CharT, _Traits >, 1405
    std::basic_ostream< _CharT, _Traits >, 1436
    std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1470
    std::basic_stringstream< _CharT, _Traits, _Alloc >, 1754
    std::ctype< _CharT >, 1956
    std::ctype< char >, 1985–1987
    std::ctype< wchar_t >, 2000, 2001
    std::ctype_byname< _CharT >, 2015
    std::ctype_byname< char >, 2038, 2039
width
    std::basic_fstream< _CharT, _Traits >, 1177
    std::basic_ifstream< _CharT, _Traits >, 1221
    std::basic_ios< _CharT, _Traits >, 1245, 1246
    std::basic_iostream< _CharT, _Traits >, 1292
    std::basic_istream< _CharT, _Traits >, 1332
    std::basic_istringstream< _CharT, _Traits, _Alloc >, 1372, 1373
    std::basic_ofstream< _CharT, _Traits >, 1405
    std::basic_ostream< _CharT, _Traits >, 1437
    std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1470
    std::basic_stringstream< _CharT, _Traits, _Alloc >, 1754
    std::ios_base, 2296
wifstream
    I/O, 170
wios
    I/O, 170
wiostream
    I/O, 170
wistream
    I/O, 170
wistringstream
    I/O, 170
wofstream
    I/O, 171
workstealing.h, 3674
wostream
    I/O, 171
wostringstream
    I/O, 171

wregex
Regular Expressions, 259
write
std::basic_fstream< _CharT, _Traits >, 1178
std::basic_iostream< _CharT, _Traits >, 1292
std::basic_ofstream< _CharT, _Traits >, 1406
std::basic_ostream< _CharT, _Traits >, 1437
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1471
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1755
ws
std, 649
wsregex_token_iterator
Regular Expressions, 259
wssub_match
Regular Expressions, 260
wstreambuf
I/O, 171
wstreampos
std, 589
wstring
Strings, 279
wstring_convert
std::wstring_convert< _Codecvt, _Elem, _Wide_alloc, _Byte_alloc >, 3272
wstringbuf
I/O, 171
wstringstream
I/O, 171

xalloc
std::basic_fstream< _CharT, _Traits >, 1178
std::basic_ifstream< _CharT, _Traits >, 1221
std::basic_ios< _CharT, _Traits >, 1246
std::basic_iostream< _CharT, _Traits >, 1293
std::basic_istream< _CharT, _Traits >, 1332
std::basic_istringstream< _CharT, _Traits, _Alloc >, 1373
std::basic_ofstream< _CharT, _Traits >, 1406
std::basic_ostream< _CharT, _Traits >, 1438
std::basic_ostringstream< _CharT, _Traits, _Alloc >, 1471
std::basic_stringstream< _CharT, _Traits, _Alloc >, 1755
std::ios_base, 2296
xsgetn
__gnu_cxx::enc_filebuf< _CharT >, 2126
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2955
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2972
std::basic_filebuf< _CharT, _Traits >, 1129
std::basic_streambuf< _CharT, _Traits >, 1502
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1709
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3265
xsputn
__gnu_cxx::enc_filebuf< _CharT >, 2127
__gnu_cxx::stdio_filebuf< _CharT, _Traits >, 2955
__gnu_cxx::stdio_sync_filebuf< _CharT, _Traits >, 2973
std::basic_filebuf< _CharT, _Traits >, 1130
std::basic_streambuf< _CharT, _Traits >, 1502
std::basic_stringbuf< _CharT, _Traits, _Alloc >, 1710
std::wbuffer_convert< _Codecvt, _Elem, _Tr >, 3266

years
Time, 375
yield
std::this_thread, 726