PD ISO/IEC TS 19568:2024

BSI Standards Publication

Programming Languages — C++ Extensions for Library Fundamentals

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 19568:2024

National foreword

This Published Document is the UK implementation of ISO/IEC TS 19568:2024. It supersedes PD ISO/IEC TS 19568:2017, which is withdrawn.

The UK participation in its preparation was entrusted to Technical Committee IST/5, Programming languages, their environments and system software interfaces.

A list of organizations represented on this committee can be obtained on request to its committee manager.

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

This publication is not to be regarded as a British Standard.

© The British Standards Institution 2024
Published by BSI Standards Limited 2024

ISBN 978 0 539 25560 7

ICS 35.060

Compliance with a Published Document cannot confer immunity from legal obligations.

This Published Document was published under the authority of the Standards Policy and Strategy Committee on 31 August 2024.

Amendments/corrigenda issued since publication

Date	Text affected

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 19568:2024

Technical Specification

ISO/IEC TS 19568

Third edition
2024-08

Programming Languages — C++ Extensions for Library Fundamentals

Langages de programmation — Extensions C++ pour la bibliothèque fondamentaux

Reference number
ISO/IEC TS 19568:2024(en)

© ISO/IEC 2024

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 19568:2024

COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2024

All rights reserved. Unless otherwise specified, or required in the context of its implementation, no part of this publication may be reproduced or utilized otherwise in any form or by any means, electronic or mechanical, including photocopying, or posting on the internet or an intranet, without prior written permission. Permission can be requested from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
CP 401 • Ch. de Blandonnet 8
CH-1214 Vernier, Geneva
Phone: +41 22 749 01 11
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Documents
Document Sharing Hub

Full document details: PD ISO/IEC TS 19568:2024

Contents

Foreword . . . . . **vi**
Introduction . . . . . **viii**
1 Scope **1**
2 Normative references **2**
3 Terms and definitions **3**
4 General principles **4**
4.1 Namespaces, headers, and modifications to standard classes 4
4.2 Feature-testing recommendations 5
5 Modifications to the C++ Standard Library **7**
5.1 General 7
5.2 Exception requirements 7

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

6 General utilities library . . . 8
6.1 Constness propagation . . . 8
6.1.1 Header <experimental/propagate_const> synopsis . . . 8
6.1.2 Class template propagate_const . . . 10
6.1.2.1 Overview . . . 10
6.1.2.2 General requirements on T . . . 12
6.1.2.3 Requirements on class type T . . . 12
6.1.2.4 Constructors . . . 13
6.1.2.5 Assignment . . . 13
6.1.2.6 Const observers . . . 14
6.1.2.7 Non-const observers . . . 14
6.1.2.8 Modifiers . . . 15
6.1.2.9 Relational operators . . . 15
6.1.2.10 Specialized algorithms . . . 18
6.1.2.11 Underlying pointer access . . . 18
6.1.2.12 Hash support . . . 18
6.1.2.13 Comparison function objects . . . 19
6.2 Scope guard support . . . 20
6.2.1 Header <experimental/scope> synopsis . . . 20
6.2.2 Class templates scope_exit, scope_fail, and scope_success . . . 21
6.2.3 Class template unique_resource . . . 24
6.2.3.1 Overview . . . 24
6.2.3.2 Constructors . . . 26
6.2.3.3 Destructor . . . 27
6.2.3.4 Assignment . . . 27
6.2.3.5 Other member functions . . . 29
6.2.3.6 unique_resource creation . . . 30
6.3 Metaprogramming and type traits . . . 31
6.3.1 Header <experimental/type_traits> synopsis . . . 31
6.3.2 Other type transformations . . . 32
6.3.3 Detection idiom . . . 34
7 Function objects . . . 36
7.1 Header <experimental/functional> synopsis . . . 36
7.2 Class template function . . . 36
7.2.1 Overview . . . 36
7.2.2 Construct/copy/destroy . . . 38
7.2.3 Modifiers . . . 39
7.2.4 Observers . . . 39

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

8 Memory . . . 40
8.1 Header <experimental/memory> synopsis . . . 40
8.2 Non-owning (observer) pointers . . . 41
8.2.1 Class template observer_ptr overview . . . 41
8.2.2 observer_ptr constructors . . . 42
8.2.3 observer_ptr observers . . . 43
8.2.4 observer_ptr conversions . . . 43
8.2.5 observer_ptr modifiers . . . 43
8.2.6 observer_ptr specialized algorithms . . . 44
8.2.7 observer_ptr hash support . . . 45
8.3 Header <experimental/memory_resource> synopsis . . . 45
8.4 Alias template resource_adaptor . . . 45
8.4.1 resource_adaptor . . . 45
8.4.2 resource_adaptor_imp constructors . . . 47
8.4.3 resource_adaptor_imp member functions . . . 47
9 Iterators library . . . 48
9.1 Header <experimental/iterator> synopsis . . . 48
9.2 Class template ostream_joiner . . . 48
9.2.1 Overview . . . 48
9.2.2 Constructor . . . 49
9.2.3 Operations . . . 50
9.2.4 Creation function . . . 50
10 Algorithms library . . . 51
10.1 Header <experimental/algorithm> synopsis . . . 51
10.2 Sampling . . . 51
10.3 Shuffle . . . 52
11 Numerics library . . . 53
11.1 Random number generation . . . 53
11.1.1 Header <experimental/random> synopsis . . . 53
11.1.2 Function template randint . . . 53

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

Foreword

ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are members of ISO or IEC participate in the development of International Standards through technical committees established by the respective organization to deal with particular fields of technical activity. ISO and IEC technical committees collaborate in fields of mutual interest. Other international organizations, governmental and non-governmental, in liaison with ISO and IEC, also take part in the work.

The procedures used to develop this document and those intended for its further maintenance are described in the ISO/IEC Directives, Part 1. In particular, the different approval criteria needed for the different types of document should be noted. This document was drafted in accordance with the editorial rules of the ISO/IEC Directives, Part 2 (see www.iso.org/directives or www.iec.ch/members_experts/refdocs).

ISO and IEC draw attention to the possibility that the implementation of this document may involve the use of (a) patent(s). ISO and IEC take no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, ISO and IEC had not received notice of (a) patent(s) which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at www.iso.org/patents and https://patents.iec.ch. ISO and IEC shall not be held responsible for identifying any or all such patent rights.

Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.

For an explanation of the voluntary nature of standards, the meaning of ISO specific terms and expressions related to conformity assessment, as well as information about ISO's adherence to the World Trade Organization (WTO) principles in the Technical Barriers to Trade (TBT) see www.iso.org/iso/foreword.html. In the IEC, see www.iec.ch/understanding-standards.

This document was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.

This third edition cancels and replaces the second edition (ISO/IEC TS 19568:2017), which has been technically revised.

The main changes are as follows:

— The document now refers to the C++ language as defined in ISO/IEC 14882:2020; the previous edition referred to ISO/IEC 14882:2017.
— Removal of features that have been added to ISO/IEC 14882: tuple utilities, logical

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

operator traits, rational arithmetic, time utilities, error support, searchers, `not_fn`, `optional`, `any`, `string_view`, shared-ownership pointers, `memory_resource`, search algorithm, numeric operations (gcd/lcm), `source_location`.

— New feature: scope guard class templates for guard types that perform automatic actions on scope exit.

— Feature modification: type-erasing classes now use `polymorphic_allocator<>`.

Any feedback or questions on this document should be directed to the user's national standards body. A complete listing of these bodies can be found at www.iso.org/members.html and www.iec.ch/national-committees.

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

Introduction [introduction]

In this document, the phrase *C++ Standard Library* refers to the library described in ISO/IEC 14882:2020, clauses 16–32.

Clauses and subclauses in this document are annotated with a so-called stable name, presented in square brackets next to the (sub)clause heading (such as "[introduction]" for this clause). Stable names aid in the discussion and evolution of this document by serving as stable references to subclauses across editions that are unaffected by changes of subclause numbering.

In addition to the main font for the document body, this document uses `upright monospace font` to display C++ source code, some of which forms part of the normative specification verbatim, *`italic monospace font`* for placeholders within source code that necessary for the specification, but whose spelling is not significant, and *ItalicSerifCamelCase* for certain concepts (comprising syntactic and semantic constraints) from the C++ Standard Library.

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

Programming languages —
C++ Extensions for Library Fundamentals

1 Scope [general.scope]

This document describes extensions to the C++ Standard Library (2). These extensions are classes and functions that are likely to be used widely within a program and/or on the interface boundaries between libraries written by different organizations.

It is intended that some of the library components be considered for standardization in a future version of C++. At present, they are not part of any C++ standard.

The goal of this document is to build more widespread existing practice for an expanded C++ standard library. It gives advice on extensions to those vendors who wish to provide them.

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

2 Normative references [general.references]

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

— ISO/IEC 14882:2020, *Programming Languages — C++*

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

3 Terms and definitions [general.terms]

For the purposes of this document, the terms and definitions given in ISO/IEC 14882:2020 apply.

ISO and IEC maintain terminology databases for use in standardization at the following addresses:

— ISO Online browsing platform: available at https://www.iso.org/obp
— IEC Electropedia: available at https://www.electropedia.org/

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

4 General principles [general]

4.1 Namespaces, headers, and modifications to standard classes [general.namespaces]

Since the extensions described in this document are experimental and not part of the C++ standard library, they should not be declared directly within namespace `std`. Unless otherwise specified, all components described in this document either:

— modify an existing interface in the C++ Standard Library in-place,
— are declared in a namespace whose name appends `::experimental::fundamentals_v3` to a namespace defined in the C++ Standard Library, such as `std` or `std::chrono`, or
— are declared in a subnamespace of a namespace described in the previous bullet, whose name is not the same as an existing subnamespace of namespace `std`.

EXAMPLE This document does not define `std::experimental::fundamentals_v3::pmr` because the C++ Standard Library defines `std::pmr`.

Each header described in this document shall import the contents of `std::experimental::fundamentals_v3` into `std::experimental` as if by

```
namespace std::experimental::inline fundamentals_v3 {}
```

This document also describes some experimental modifications to existing interfaces in the C++ Standard Library.

Unless otherwise specified, references to other entities described in this document are assumed to be qualified with `std::experimental::fundamentals_v3::`, and references to entities described in the standard are assumed to be qualified with `std::`.

Extensions that are expected to eventually be added to an existing header `<meow>` are provided inside the `<experimental/meow>` header, which shall include the standard contents of `<meow>` as if by

```
#include <meow>
```

New headers are also provided in the `<experimental/>` directory, but without such an `#include`. Table 1 lists the headers that are specified by this document.

Table 1 — C++ library headers

`<experimental/algorithm>`	`<experimental/memory>`	`<experimental/scope>`
`<experimental/functional>`	`<experimental/memory_resource>`	`<experimental/type_traits>`
`<experimental/future>`	`<experimental/propagate_const>`	`<experimental/utility>`
`<experimental/iterator>`	`<experimental/random>`	

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

NOTE This is the last in a series of revisions of this document planned by the C++ committee; while there are no plans to resume the series, any future versions will define their contents in `std::experimental::fundamentals_v4`, `std::experimental::fundamentals_v5`, etc., with the most recent implemented version inlined into `std::experimental`.

4.2 Feature-testing recommendations [general.feature.test]

For the sake of improved portability between partial implementations of various C++ standards, implementers and programmers are recommended to follow the guidelines in this subclause concerning feature-test macros.

Implementers who provide a new standard feature should define a macro with the recommended name, in the same circumstances under which the feature is available (for example, taking into account relevant command-line options), to indicate the presence of support for that feature. Implementers should define that macro with the value specified in the most recent version of this document that they have implemented. The recommended macro name is "`__cpp_lib_experimental_`" followed by the string in the "Macro Name Suffix" column. Table 2 lists the headers and recommended feature-test macros for the features specified in this document.

Programmers who wish to determine whether a feature is available in an implementation should base that determination on the presence of the header (determined with `__has_include(<header/name>)`) and the state of the macro with the recommended name. (The absence of a tested feature may result in a program with decreased functionality, or the relevant functionality may be provided in a different way. A program that strictly depends on support for a feature can just try to use the feature unconditionally; presumably, on an implementation lacking necessary support, translation will fail.)

Table 2 — Significant features in this document

Feature	Primary Subclause	Macro Name Suffix	Value	Header
Const-propagating wrapper	6.1	`propagate_const`	201505	`<experimental/propagate_const>`
Generic scope guard and RAII wrapper	6.2	`scope`	201902	`<experimental/scope>`
Invocation type traits	6.3.2	`invocation_type`	201406	`<experimental/type_traits>`
Detection metaprograms	6.3.3	`detect`	201505	`<experimental/type_traits>`

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

Feature	Primary Subclause	Macro Name Suffix	Value	Header
Polymorphic allocator for `std::function`	7.2	`function_polymorphic_allocator`	202211	`<experimental/functional>`
Polymorphic memory resources	8.3	`memory_resources`	201803	`<experimental/memory_resouce>`
Non-owning pointer wrapper	8.2	`observer_ptr`	201411	`<experimental/memory>`
Delimited iterators	9.2	`ostream_joiner`	201411	`<experimental/iterator>`
Random sampling	10.2	`sample`	201402	`<experimental/algorithm>`
Replacement for `std::rand`	11.1.2	`randint`	201511	`<experimental/random>`

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

5 Modifications to the C++ Standard Library [mods]

5.1 General [mods.general]

Implementations that conform to this document shall behave as if the modifications contained in this clause are made to ISO/IEC 14882:2020.

References to clauses within ISO/IEC 14882:2020 are written as "C++20, §3.2".

Unless otherwise specified, the whole of the Library introduction of ISO/IEC 14882:2020 (C++20, §16) is included into this document by reference.

5.2 Exception requirements [mods.exception.requirements]

The following modifications to the Library introduction (C++20, §16) allow the destructor of `scope_success` (6.2) to throw exceptions.

The requirements of C++20, §16.4.5.8 shall apply with the following modification: An applicable *Throws:* paragraph (in addition to any applicable *Required behavior:* paragraph) can allow a replacement function or handler function or destructor operation to exit via an exception.

The requirements of C++20, §16.4.6.13 shall apply with the following modifications: Destructor operations defined in the C++ standard library are permitted to throw exceptions if this is explicitly specified. Only those destructors in the C++ standard library that do not have an exception specification shall behave as if they had a non-throwing exception specification.

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

6 General utilities library [utilities]

6.1 Constness propagation [propagate_const]

6.1.1 Header <experimental/propagate_const> synopsis [propagate_const.syn]

```
namespace std {
  namespace experimental::inline fundamentals_v3 {

    // 6.1.2.1, Overview
    template <class T> class propagate_const;

    // 6.1.2.9, Relational operators
    template <class T>
      constexpr bool operator==(const propagate_const<T>& pt, nullptr_t);
    template <class T>
      constexpr bool operator==(nullptr_t, const propagate_const<T>& pu);

    template <class T>
      constexpr bool operator!=(const propagate_const<T>& pt, nullptr_t);
    template <class T>
      constexpr bool operator!=(nullptr_t, const propagate_const<T>& pu);

    template <class T, class U>
      constexpr bool operator==(const propagate_const<T>& pt,
                                const propagate_const<U>& pu);
    template <class T, class U>
      constexpr bool operator!=(const propagate_const<T>& pt,
                                const propagate_const<U>& pu);
    template <class T, class U>
      constexpr bool operator<(const propagate_const<T>& pt,
                                const propagate_const<U>& pu);
    template <class T, class U>
      constexpr bool operator>(const propagate_const<T>& pt,
                                const propagate_const<U>& pu);
    template <class T, class U>
      constexpr bool operator<=(const propagate_const<T>& pt,
                                const propagate_const<U>& pu);
```

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

```
template <class T, class U>
  constexpr bool operator>=(const propagate_const<T>& pt,
                            const propagate_const<U>& pu);

template <class T, class U>
  constexpr bool operator==(const propagate_const<T>& pt, const U& u);
template <class T, class U>
  constexpr bool operator!=(const propagate_const<T>& pt, const U& u);
template <class T, class U>
  constexpr bool operator<(const propagate_const<T>& pt, const U& u);
template <class T, class U>
  constexpr bool operator>(const propagate_const<T>& pt, const U& u);
template <class T, class U>
  constexpr bool operator<=(const propagate_const<T>& pt, const U& u);
template <class T, class U>
  constexpr bool operator>=(const propagate_const<T>& pt, const U& u);

template <class T, class U>
  constexpr bool operator==(const T& t, const propagate_const<U>& pu);
template <class T, class U>
  constexpr bool operator!=(const T& t, const propagate_const<U>& pu);
template <class T, class U>
  constexpr bool operator<(const T& t, const propagate_const<U>& pu);
template <class T, class U>
  constexpr bool operator>(const T& t, const propagate_const<U>& pu);
template <class T, class U>
  constexpr bool operator<=(const T& t, const propagate_const<U>& pu);
template <class T, class U>
  constexpr bool operator>=(const T& t, const propagate_const<U>& pu);

// 6.1.2.10, Specialized algorithms
template <class T>
  constexpr void swap(propagate_const<T>& pt,
                      propagate_const<T>& pt2) noexcept(see below);

// 6.1.2.11, Underlying pointer access
template <class T>
  constexpr const T& get_underlying(const propagate_const<T>& pt) noexcept;
```

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024

```
    template <class T>
      constexpr T& get_underlying(propagate_const<T>& pt) noexcept;

  } // namespace experimental::inline fundamentals_v3

  // 6.1.2.12, Hash support
  template <class T> struct hash;
  template <class T>
    struct hash<experimental::fundamentals_v3::propagate_const<T>>;

  // 6.1.2.13, Comparison function objects
  template <class T> struct equal_to;
  template <class T>
    struct equal_to<experimental::fundamentals_v3::propagate_const<T>>;
  template <class T> struct not_equal_to;
  template <class T>
    struct not_equal_to<experimental::fundamentals_v3::propagate_const<T>>;
  template <class T> struct less;
  template <class T>
    struct less<experimental::fundamentals_v3::propagate_const<T>>;
  template <class T> struct greater;
  template <class T>
    struct greater<experimental::fundamentals_v3::propagate_const<T>>;
  template <class T> struct less_equal;
  template <class T>
    struct less_equal<experimental::fundamentals_v3::propagate_const<T>>;
  template <class T> struct greater_equal;
  template <class T>
    struct greater_equal<experimental::fundamentals_v3::propagate_const<T>>;

} // namespace std
```

6.1.2 Class template propagate_const [propagate_const.tmpl]

6.1.2.1 Overview [propagate_const.overview]

```
namespace std::experimental::inline fundamentals_v3 {

  template <class T> class propagate_const {
```

© ISO/IEC 2024 – All rights reserved

Documents
Document Sharing Hub
Full document details: PD ISO/IEC TS 19568:2024