Technical Specification

ISO/IEC TS 19568

Programming Languages — C++ Extensions for Library Fundamentals

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

Third edition
2024-08

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

Contents

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 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.

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.

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.

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++*

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/

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>`	

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>`

Feature	Primary Subclause	Macro Name Suffix	Value	Header
Polymorphic allocator for `std::function`	7.2	`function_polymor-phic_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>`