INTERNATIONAL STANDARD

ISO/IEC 14882

Sixth edition
2020-12

Programming languages — C++

Langages de programmation — C++

Reference number
ISO/IEC 14882:2020(E)

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

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).

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. ISO and IEC shall not be held responsible for identifying any or all such patent rights. Details of any patent rights identified during the development of the document will be in the Introduction and/or on the ISO list of patent declarations received (see www.iso.org/patents) or the IEC list of patent declarations received (see http://patents.iec.ch).

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

For an explanation on 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.

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 sixth edition cancels and replaces the fifth edition (ISO/IEC 14882:2017), which has been technically revised.

The main changes compared to the previous edition are as follows:

- inclusion of the provisions of ISO/IEC TS 19217:2015, ISO/IEC TS 21425:2017, ISO/IEC TS 22277:2017, ISO/IEC TS 21544:2018, portions of ISO/IEC TS 19571:2016, and portions of ISO/IEC TS 19568:2017.
- addition of concepts, *requires-clause*s, *requires-expression*s, and `<concepts>` (18.3) header
- addition of coroutines, including `co_yield`, `co_await`, and `co_return` keywords and `<coroutine>` (17.12.2) header
- addition of modules, *import-declaration*s, and *export-declaration*s
- addition of three-way comparison, defaulted comparisons, rewriting of comparison operator expressions, and `<compare>` (17.11.1) header
- addition of designated initializers
- support for class types and floating-point types as the type of a non-type template parameter
- new attributes `[[no_unique_address]]`, `[[likely]]`, `[[unlikely]]`
- support for optional reason string in `[[nodiscard]]` attribute
- ability to require constant initialization with `constinit` keyword
- ability to require constant evaluation with `consteval` keyword
- extensions to constant evaluation
- support for controlling destruction in a class-specific operator delete function
- addition of `using enum` declaration
- addition of `char8_t` type
- support for an initializer statement in range-based for loops

— support for default member initializers for bit-fields
— support for parenthesized aggregate initialization
— extensions to lambda expressions
— extensions to structured bindings
— support for inline namespaces in nested namespace definitions
— support for conditionally-explicit member functions
— extensions to class template argument deduction
— reduced cases in which `typename` is required
— support for calling an undeclared *template-id* via argument-dependent name lookup
— revised memory model
— extended support for variadic macros with `__VA_OPT__`
— feature test macros and `<version>` (17.3.2) header
— addition of ranges and `<ranges>` (24.2) header
— addition of calendar and time zone support
— addition of text formatting library and `<format>` (20.20.1) header
— addition of `<barrier>` (32.8.3.2), `<latch>` (32.8.2.2), and `<semaphore>` (32.7.2) headers
— addition of mathematical constants library and `<numbers>` (26.9.1) header
— support for representing source locations and `<source_location>` (17.8.1) header
— addition of `span` view and `<span>` (22.7.2) header
— addition of joining thread class and `<stop_token>` (32.3.2) header
— extensions to atomic types and operations
— addition of `unsequenced` execution policy
— new utility functions, types, and templates in the standard library
— addition of bit manipulation library and `<bit>` (26.5.2) header
— addition of a synchronized buffered output stream and `<syncstream>` (29.10.1) header
— support for heterogeneous lookup for unordered containers
— support for element existence detection in associative containers
— support for move semantics in `<numeric>` (25.9) algorithms
— support for efficient access to the buffer of a `basic_stringbuf`
— extended constant expression evaluation support in the standard library

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.

1 Scope [intro.scope]

1 This document specifies requirements for implementations of the C++ programming language. The first such requirement is that they implement the language, so this document also defines C++. Other requirements and relaxations of the first requirement appear at various places within this document.

2 C++ is a general purpose programming language based on the C programming language as described in ISO/IEC 9899:2018 *Programming languages — C* (hereinafter referred to as the *C standard*). C++ provides many facilities beyond those provided by C, including additional data types, classes, templates, exceptions, namespaces, operator overloading, function name overloading, references, free store management operators, and additional library facilities.

2 Normative references [intro.refs]

1 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.

(1.1) — ISO/IEC 2382, *Information technology — Vocabulary*

(1.2) — ISO 8601:2004, *Data elements and interchange formats — Information interchange — Representation of dates and times*

(1.3) — ISO/IEC 9899:2018, *Programming languages — C*

(1.4) — ISO/IEC 9945:2003, *Information Technology — Portable Operating System Interface (POSIX*[1]*)*

(1.5) — ISO/IEC 10646, *Information technology — Universal Coded Character Set (UCS)*

(1.6) — ISO/IEC 10646:2003,[2] *Information technology — Universal Multiple-Octet Coded Character Set (UCS)*

(1.7) — ISO 80000-2:2009, *Quantities and units — Part 2: Mathematical signs and symbols to be used in the natural sciences and technology*

(1.8) — Ecma International, *ECMAScript*[3] *Language Specification*, Standard Ecma-262, third edition, 1999.

2 The library described in ISO/IEC 9899:2018, Clause 7, is hereinafter called the *C standard library.*[4]

3 The operating system interface described in ISO/IEC 9945:2003 is hereinafter called *POSIX.*

4 The ECMAScript Language Specification described in Standard Ecma-262 is hereinafter called *ECMA-262.*

5 [*Note 1*: References to ISO/IEC 10646:2003 are used only to support deprecated features (D.21). — *end note*]

1) POSIX® is a registered trademark of the Institute of Electrical and Electronic Engineers, Inc. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of this product.

2) Cancelled and replaced by ISO/IEC 10646:2017.

3) ECMAScript® is a registered trademark of Ecma International. This information is given for the convenience of users of this document and does not constitute an endorsement by ISO or IEC of this product.

4) With the qualifications noted in Clause 17 through Clause 32 and in C.6, the C standard library is a subset of the C++ standard library.

3 Terms and definitions [intro.defs]

1 For the purposes of this document, the terms and definitions given in ISO/IEC 2382, the terms, definitions, and symbols given in ISO 80000-2:2009, and the following apply.

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

(2.1) — ISO Online browsing platform: available at https://www.iso.org/obp

(2.2) — IEC Electropedia: available at http://www.electropedia.org/

3 Terms that are used only in a small portion of this document are defined where they are used and italicized where they are defined.

3.1 [defns.access]

access

⟨execution-time action⟩ read or modify the value of an object

[*Note 1 to entry*: Only objects of scalar type can be accessed. Reads of scalar objects are described in 7.3.2 and modifications of scalar objects are described in 7.6.19, 7.6.1.6, and 7.6.2.3. Attempts to read or modify an object of class type typically invoke a constructor (11.4.5) or assignment operator (11.4.6); such invocations do not themselves constitute accesses, although they may involve accesses of scalar subobjects. — *end note*]

3.2 [defns.arbitrary.stream]

arbitrary-positional stream

⟨library⟩ stream that can seek to any integral position within the length of the stream

[*Note 1 to entry*: Every arbitrary-positional stream is also a repositional stream (3.42). — *end note*]

3.3 [defns.argument]

argument

⟨function call expression⟩ expression in the comma-separated list bounded by the parentheses

3.4 [defns.argument.macro]

argument

⟨function-like macro⟩ sequence of preprocessing tokens in the comma-separated list bounded by the parentheses

3.5 [defns.argument.throw]

argument

⟨throw expression⟩ operand of `throw`

3.6 [defns.argument.templ]

argument

⟨template instantiation⟩ *constant-expression*, *type-id*, or *id-expression* in the comma-separated list bounded by the angle brackets

3.7 [defns.block]

block

⟨execution⟩ wait for some condition (other than for the implementation to execute the execution steps of the thread of execution) to be satisfied before continuing execution past the blocking operation

3.8 [defns.block.stmt]

block

⟨statement⟩ compound statement

3.9 [defns.character]

character

⟨library⟩ object which, when treated sequentially, can represent text

[*Note 1 to entry*: The term does not mean only `char`, `char8_t`, `char16_t`, `char32_t`, and `wchar_t` objects (6.8.2), but any value that can be represented by a type that provides the definitions specified in Clause 21, Clause 28, Clause 29, or Clause 30. — *end note*]

