
**Information technology —
Programming languages — C**

Technologies de l'information — Langages de programmation — C

iTeh STANDARD PREVIEW
(standards.iteh.ai)

[ISO/IEC 9899:2018](https://standards.iteh.ai/catalog/standards/sist/1affe60f-8060-47f1-89c0-c6856c6eab89/iso-iec-9899-2018)

<https://standards.iteh.ai/catalog/standards/sist/1affe60f-8060-47f1-89c0-c6856c6eab89/iso-iec-9899-2018>



Reference number
ISO/IEC 9899:2018(E)

© ISO/IEC 2018

iTeh STANDARD PREVIEW (standards.iteh.ai)

ISO/IEC 9899:2018

<https://standards.iteh.ai/catalog/standards/sist/1affe60f-8060-47f1-89c0-c6856c6eab89/iso-iec-9899-2018>



COPYRIGHT PROTECTED DOCUMENT

© ISO/IEC 2018

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
Fax: +41 22 749 09 47
Email: copyright@iso.org
Website: www.iso.org

Published in Switzerland

Contents

Foreword	xi
Introduction	xii
1 Scope	1
2 Normative references	2
3 Terms, definitions and symbols	3
4 Conformance	8
5 Environment	9
5.1 Conceptual models	9
5.1.1 Translation environment	9
5.1.2 Execution environments	10
5.2 Environmental considerations	17
5.2.1 Character sets	17
5.2.2 Character display semantics	19
5.2.3 Signals and interrupts	19
5.2.4 Environmental limits	19
6 Language	28
6.1 Notation	28
6.2 Concepts	28
6.2.1 Scopes of identifiers	28
6.2.2 Linkages of identifiers	29
6.2.3 Name spaces of identifiers	29
6.2.4 Storage durations of objects	30
6.2.5 Types	31
6.2.6 Representations of types	33
6.2.7 Compatible type and composite type	35
6.2.8 Alignment of objects	36
6.3 Conversions	37
6.3.1 Arithmetic operands	37
6.3.2 Other operands	40
6.4 Lexical elements	41
6.4.1 Keywords	42
6.4.2 Identifiers	43

6.4.3	Universal character names	44
6.4.4	Constants	45
6.4.5	String literals	50
6.4.6	Punctuators	52
6.4.7	Header names	53
6.4.8	Preprocessing numbers	53
6.4.9	Comments	54
6.5	Expressions	55
6.5.1	Primary expressions	56
6.5.2	Postfix operators	57
6.5.3	Unary operators	63
6.5.4	Cast operators	65
6.5.5	Multiplicative operators	66
6.5.6	Additive operators	66
6.5.7	Bitwise shift operators	68
6.5.8	Relational operators	68
6.5.9	Equality operators	69
6.5.10	Bitwise AND operator	70
6.5.11	Bitwise exclusive OR operator	70
6.5.12	Bitwise inclusive OR operator	70
6.5.13	Logical AND operator	71
6.5.14	Logical OR operator	71
6.5.15	Conditional operator	71
6.5.16	Assignment operators	72
6.5.17	Comma operator	75
6.6	Constant expressions	76
6.7	Declarations	78
6.7.1	Storage-class specifiers	79
6.7.2	Type specifiers	79
6.7.3	Type qualifiers	87
6.7.4	Function specifiers	90
6.7.5	Alignment specifier	92
6.7.6	Declarators	92
6.7.7	Type names	98
6.7.8	Type definitions	99
6.7.9	Initialization	100
6.7.10	Static assertions	105
6.8	Statements and blocks	106
6.8.1	Labeled statements	106
6.8.2	Compound statement	107

6.8.3	Expression and null statements	107
6.8.4	Selection statements	108
6.8.5	Iteration statements	109
6.8.6	Jump statements	110
6.9	External definitions	113
6.9.1	Function definitions	113
6.9.2	External object definitions	115
6.10	Preprocessing directives	117
6.10.1	Conditional inclusion	118
6.10.2	Source file inclusion	119
6.10.3	Macro replacement	121
6.10.4	Line control	126
6.10.5	Error directive	126
6.10.6	Pragma directive	127
6.10.7	Null directive	127
6.10.8	Predefined macro names	127
6.10.9	Pragma operator	129
6.11	Future language directions	130
6.11.1	Floating types	130
6.11.2	Linkages of identifiers	130
6.11.3	External names	130
6.11.4	Character escape sequences	130
6.11.5	Storage-class specifiers	130
6.11.6	Function declarators	130
6.11.7	Function definitions	130
6.11.8	Pragma directives	130
6.11.9	Predefined macro names	130
7	Library	131
7.1	Introduction	131
7.1.1	Definitions of terms	131
7.1.2	Standard headers	131
7.1.3	Reserved identifiers	132
7.1.4	Use of library functions	132
7.2	Diagnostics <assert.h>	135
7.2.1	Program diagnostics	135
7.3	Complex arithmetic <complex.h>	136
7.3.1	Introduction	136
7.3.2	Conventions	136
7.3.3	Branch cuts	136

7.3.4	The CX_LIMITED_RANGE pragma	137
7.3.5	Trigonometric functions	137
7.3.6	Hyperbolic functions	139
7.3.7	Exponential and logarithmic functions	140
7.3.8	Power and absolute-value functions	141
7.3.9	Manipulation functions	142
7.4	Character handling <ctype.h>	145
7.4.1	Character classification functions	145
7.4.2	Character case mapping functions	147
7.5	Errors <errno.h>	149
7.6	Floating-point environment <fenv.h>	150
7.6.1	The FENV_ACCESS pragma	151
7.6.2	Floating-point exceptions	152
7.6.3	Rounding	154
7.6.4	Environment	155
7.7	Characteristics of floating types <float.h>	157
7.8	Format conversion of integer types <inttypes.h>	158
7.8.1	Macros for format specifiers	158
7.8.2	Functions for greatest-width integer types	159
7.9	Alternative spellings <iso646.h>	161
7.10	Sizes of integer types <limits.h>	162
7.11	Localization <locale.h>	163
7.11.1	Locale control	163
7.11.2	Numeric formatting convention inquiry	164
7.12	Mathematics <math.h>	169
7.12.1	Treatment of error conditions	170
7.12.2	The FP_CONTRACT pragma	171
7.12.3	Classification macros	172
7.12.4	Trigonometric functions	173
7.12.5	Hyperbolic functions	175
7.12.6	Exponential and logarithmic functions	177
7.12.7	Power and absolute-value functions	180
7.12.8	Error and gamma functions	182
7.12.9	Nearest integer functions	183
7.12.10	Remainder functions	185
7.12.11	Manipulation functions	186
7.12.12	Maximum, minimum, and positive difference functions	187
7.12.13	Floating multiply-add	188
7.12.14	Comparison macros	189
7.13	Nonlocal jumps <setjmp.h>	191

7.13.1	Save calling environment	191
7.13.2	Restore calling environment	191
7.14	Signal handling <signal.h>	193
7.14.1	Specify signal handling	193
7.14.2	Send signal	194
7.15	Alignment <stdalign.h>	196
7.16	Variable arguments <stdarg.h>	197
7.16.1	Variable argument list access macros	197
7.17	Atomics <stdatomic.h>	200
7.17.1	Introduction	200
7.17.2	Initialization	201
7.17.3	Order and consistency	201
7.17.4	Fences	204
7.17.5	Lock-free property	205
7.17.6	Atomic integer types	205
7.17.7	Operations on atomic types	206
7.17.8	Atomic flag type and operations	208
7.18	Boolean type and values <stdbool.h>	210
7.19	Common definitions <stddef.h>	211
7.20	Integer types <stdint.h>	212
7.20.1	Integer types	212
7.20.2	Limits of specified-width integer types	213
7.20.3	Limits of other integer types	215
7.20.4	Macros for integer constants	216
7.21	Input/output <stdio.h>	217
7.21.1	Introduction	217
7.21.2	Streams	218
7.21.3	Files	219
7.21.4	Operations on files	221
7.21.5	File access functions	222
7.21.6	Formatted input/output functions	225
7.21.7	Character input/output functions	241
7.21.8	Direct input/output functions	244
7.21.9	File positioning functions	245
7.21.10	Error-handling functions	247
7.22	General utilities <stdlib.h>	249
7.22.1	Numeric conversion functions	249
7.22.2	Pseudo-random sequence generation functions	253
7.22.3	Memory management functions	254
7.22.4	Communication with the environment	256

7.22.5	Searching and sorting utilities	259
7.22.6	Integer arithmetic functions	260
7.22.7	Multibyte/wide character conversion functions	261
7.22.8	Multibyte/wide string conversion functions	262
7.23	_Noreturn <stdnoreturn.h>	264
7.24	String handling <string.h>	265
7.24.1	String function conventions	265
7.24.2	Copying functions	265
7.24.3	Concatenation functions	266
7.24.4	Comparison functions	267
7.24.5	Search functions	268
7.24.6	Miscellaneous functions	271
7.25	Type-generic math <tgmath.h>	273
7.26	Threads <threads.h>	275
7.26.1	Introduction	275
7.26.2	Initialization functions	276
7.26.3	Condition variable functions	276
7.26.4	Mutex functions	278
7.26.5	Thread functions	280
7.26.6	Thread-specific storage functions	282
7.27	Date and time <time.h>	285
7.27.1	Components of time	285
7.27.2	Time manipulation functions	286
7.27.3	Time conversion functions	288
7.28	Unicode utilities <uchar.h>	293
7.28.1	Restartable multibyte/wide character conversion functions	293
7.29	Extended multibyte and wide character utilities <wchar.h>	296
7.29.1	Introduction	296
7.29.2	Formatted wide character input/output functions	296
7.29.3	Wide character input/output functions	308
7.29.4	General wide string utilities	312
7.29.4.1	Wide string numeric conversion functions	312
7.29.4.2	Wide string copying functions	315
7.29.4.3	Wide string concatenation functions	316
7.29.4.4	Wide string comparison functions	316
7.29.4.5	Wide string search functions	318
7.29.4.6	Miscellaneous functions	321
7.29.5	Wide character time conversion functions	321
7.29.6	Extended multibyte/wide character conversion utilities	322
7.29.6.1	Single-byte/wide character conversion functions	322

7.29.6.2	Conversion state functions	323
7.29.6.3	Restartable multibyte/wide character conversion functions	323
7.29.6.4	Restartable multibyte/wide string conversion functions	325
7.30	Wide character classification and mapping utilities <wctype.h>	327
7.30.1	Introduction	327
7.30.2	Wide character classification utilities	327
7.30.2.1	Wide character classification functions	327
7.30.2.2	Extensible wide character classification functions	330
7.30.3	Wide character case mapping utilities	331
7.30.3.1	Wide character case mapping functions	331
7.30.3.2	Extensible wide character case mapping functions	331
7.31	Future library directions	333
7.31.1	Complex arithmetic <complex.h>	333
7.31.2	Character handling <ctype.h>	333
7.31.3	Errors <errno.h>	333
7.31.4	Floating-point environment <fenv.h>	333
7.31.5	Format conversion of integer types <inttypes.h>	333
7.31.6	Localization <locale.h>	333
7.31.7	Signal handling <signal.h>	333
7.31.8	Atomics <stdatomic.h>	333
7.31.9	Boolean type and values <stdbool.h>	333
7.31.10	Integer types <stdint.h>	333
7.31.11	Input/output <stdio.h>	334
7.31.12	General utilities <stdlib.h>	334
7.31.13	String handling <string.h>	334
7.31.14	Date and time <time.h>	334
7.31.15	Threads <threads.h>	334
7.31.16	Extended multibyte and wide character utilities <wchar.h>	334
7.31.17	Wide character classification and mapping utilities <wctype.h>	334
Annex A (informative)	Language syntax summary	335
Annex B (informative)	Library summary	347
Annex C (informative)	Sequence points	367
Annex D (normative)	Universal character names for identifiers	368
Annex E (informative)	Implementation limits	369
Annex F (normative)	IEC 60559 floating-point arithmetic	371
Annex G (normative)	IEC 60559-compatible complex arithmetic	389

Annex H (informative) Language independent arithmetic	399
Annex I (informative) Common warnings	403
Annex J (informative) Portability issues	404
Annex K (normative) Bounds-checking interfaces	425
Annex L (normative) Analyzability	474
Annex M (informative) Change History	476
Bibliography	479
Index	480

iTeh STANDARD PREVIEW **(standards.iteh.ai)**

[ISO/IEC 9899:2018](https://standards.iteh.ai/catalog/standards/sist/1affe60f-8060-47f1-89c0-c6856c6eab89/iso-iec-9899-2018)

<https://standards.iteh.ai/catalog/standards/sist/1affe60f-8060-47f1-89c0-c6856c6eab89/iso-iec-9899-2018>

Foreword

- 1 ISO (the International Organization for Standardization) and IEC (the International Electrotechnical Commission) form the specialized system for worldwide standardization. National bodies that are member 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. In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1.
- 2 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).
- 3 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).
- 4 Any trade name used in this document is information given for the convenience of users and does not constitute an endorsement.
- 5 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 the following URL: www.iso.org/iso/foreword.html.
- 6 This document was prepared by Technical Committee ISO/IEC JTC 1, *Information technology*, Subcommittee SC 22, *Programming languages, their environments and system software interfaces*.
- 7 This fourth edition cancels and replaces the third edition, ISO/IEC 9899:2011, which has been technically revised. It also incorporates the Technical Corrigendum ISO/IEC 9899:2011/Cor 1:2012.
- 8 There are no major changes in this edition, only technical corrections and clarifications.
- 9 A complete change history can be found in Annex M.

Introduction

- 1 With the introduction of new devices and extended character sets, new features could be added to this document. Subclauses in the language and library clauses warn implementors and programmers of usages which, though valid in themselves, could conflict with future additions.
- 2 Certain features are *obsolescent*, which means that they could be considered for withdrawal in future revisions of this document. They are retained because of their widespread use, but their use in new implementations (for implementation features) or new programs (for language [6.11] or library features [7.31]) is discouraged.
- 3 This document is divided into four major subdivisions:
 - preliminary elements (Clauses 1–4);
 - the characteristics of environments that translate and execute C programs (Clause 5);
 - the language syntax, constraints, and semantics (Clause 6);
 - the library facilities (Clause 7).
- 4 Examples are provided to illustrate possible forms of the constructions described. Footnotes are provided to emphasize consequences of the rules described in that subclause or elsewhere in this document. References are used to refer to other related subclauses. Recommendations are provided to give advice or guidance to implementors. Annexes define optional features, provide additional information and summarize the information contained in this document. A bibliography lists documents that were referred to during the preparation of this document.
- 5 The language clause (Clause 6) is derived from “The C Reference Manual”.
- 6 The library clause (Clause 7) is based on the 1984 *usr/group Standard*.
- 7 The Working Group responsible for this document (WG 14) maintains a site on the World Wide Web at <http://www.open-std.org/JTC1/SC22/WG14/> containing ancillary information that may be of interest to some readers such as a Rationale for many of the decisions made during its preparation and a log of Defect Reports and Responses.

Programming languages — C

1. Scope

- 1 This document specifies the form and establishes the interpretation of programs written in the C programming language.¹⁾ It specifies
 - the representation of C programs;
 - the syntax and constraints of the C language;
 - the semantic rules for interpreting C programs;
 - the representation of input data to be processed by C programs;
 - the representation of output data produced by C programs;
 - the restrictions and limits imposed by a conforming implementation of C.
- 2 This document does not specify
 - the mechanism by which C programs are transformed for use by a data-processing system;
 - the mechanism by which C programs are invoked for use by a data-processing system;
 - the mechanism by which input data are transformed for use by a C program;
 - the mechanism by which output data are transformed after being produced by a C program;
 - the size or complexity of a program and its data that will exceed the capacity of any specific data-processing system or the capacity of a particular processor;
 - all minimal requirements of a data-processing system that is capable of supporting a conforming implementation.

¹⁾This document is designed to promote the portability of C programs among a variety of data-processing systems. It is intended for use by implementors and programmers. Annex J gives an overview of portability issues that a C program might encounter.

2. Normative references

- 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.
- 2 ISO/IEC 2382:2015, *Information technology — Vocabulary*. Available from the ISO online browsing platform at <http://www.iso.org/obp>
- 3 ISO 4217, *Codes for the representation of currencies and funds*
- 4 ISO 8601, *Data elements and interchange formats — Information interchange — Representation of dates and times*
- 5 ISO/IEC 10646, *Information technology — Universal Coded Character Set (UCS)*. Available from the ISO/IEC Information Technology Task Force (ITTF) web site at http://isotc.iso.org/livelink/livelink/fetch/2000/2489/Ittf_Home/PubliclyAvailableStandards.htm
- 6 IEC 60559:1989, *Binary floating-point arithmetic for microprocessor systems* (previously designated IEC 559:1989)
- 7 ISO 80000–2, *Quantities and units — Part 2: Mathematical signs and symbols to be used in the natural sciences and technology*

iTeh STANDARD PREVIEW
(standards.iteh.ai)

[ISO/IEC 9899:2018](https://standards.iteh.ai/catalog/standards/sist/1affe60f-8060-47f1-89c0-c6856c6eab89/iso-iec-9899-2018)

<https://standards.iteh.ai/catalog/standards/sist/1affe60f-8060-47f1-89c0-c6856c6eab89/iso-iec-9899-2018>

3. Terms, definitions and symbols

- 1 For the purposes of this document, the terms and definitions given in ISO/IEC 2382, ISO 80000–2, and the following apply.
- 2 ISO and IEC maintain terminological databases for use in standardization at the following addresses:
 - ISO Online browsing platform: available at <https://www.iso.org/obp>
 - IEC Electropedia: available at <http://www.electropedia.org/>
- 3 Additional terms are defined where they appear in *italic* type or on the left side of a syntax rule. Terms explicitly defined in this document are not to be presumed to refer implicitly to similar terms defined elsewhere.

3.1

- 1 **access (verb)**

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

- 2 **Note 1 to entry:** Where only one of these two actions is meant, “read” or “modify” is used.
- 3 **Note 2 to entry:** “Modify” includes the case where the new value being stored is the same as the previous value.
- 4 **Note 3 to entry:** Expressions that are not evaluated do not access objects.

3.2

- 1 **alignment**

requirement that objects of a particular type be located on storage boundaries with addresses that are particular multiples of a byte address

iTeh STANDARD PREVIEW
(standards.iteh.ai)
ISO/IEC 9899:2018
<https://standards.iteh.ai/catalog/standards/sist/1affe60f-8060-47f1-89c0-c6856c6eab89/iso-iec-9899-2018>

3.3

- 1 **argument**

actual argument

DEPRECATED: actual parameter

expression in the comma-separated list bounded by the parentheses in a function call expression, or a sequence of preprocessing tokens in the comma-separated list bounded by the parentheses in a function-like macro invocation

3.4

- 1 **behavior**

external appearance or action

3.4.1

- 1 **implementation-defined behavior**

unspecified behavior where each implementation documents how the choice is made

- 2 **Note 1 to entry:** J.3 gives an overview over properties of C programs that lead to implementation-defined behavior.
- 3 **EXAMPLE** An example of implementation-defined behavior is the propagation of the high-order bit when a signed integer is shifted right.

3.4.2

- 1 **locale-specific behavior**

behavior that depends on local conventions of nationality, culture, and language that each implementation documents