

INTERNATIONAL STANDARD

**ISO/IEC
9899**

Second edition
1999-12-01

Langages de programmation — C

iTeh STANDARD PREVIEW
(standards.iteh.ai)

[ISO/IEC 9899:1999](https://standards.iteh.ai/catalog/standards/sist/7451fd1d-bac9-4e45-b69f-72f83d536ae1/iso-iec-9899-1999)

<https://standards.iteh.ai/catalog/standards/sist/7451fd1d-bac9-4e45-b69f-72f83d536ae1/iso-iec-9899-1999>

Programing languages — C

Reference number
ISO/IEC 9899:1999(E)

© ISO/IEC 1999

This PDF file may contain embedded typefaces. In accordance with Adobe's licensing policy, this file may be printed or viewed but shall not be edited unless the typefaces which are embedded are licensed to and installed on the computer performing the editing. In downloading this file, parties accept therein the responsibility of not infringing Adobe's licensing policy. The ISO Central Secretariat accepts no liability in this area.

Adobe is a trademark of Adobe Systems Incorporated.

Details of the software products used to create this PDF file can be found in the General Info relative to the file; the PDF-creation parameters were optimized for printing. Every care has been taken to ensure that the file is suitable for use by ISO member bodies. In the unlikely event that a problem relating to it is found, please inform the Central Secretariat at the address given below.

iTeh STANDARD PREVIEW (standards.iteh.ai)

[ISO/IEC 9899:1999](#)

<https://standards.iteh.ai/catalog/standards/sist/7451fd1d-bac9-4e45-b69f-72f83d536ae1/iso-iec-9899-1999>

© ISO/IEC 1999

All rights reserved. Unless otherwise specified, no part of this publication may be reproduced or utilized in any form or by any means, electronic or mechanical, including photocopying and microfilm, without permission in writing from either ISO at the address below or ISO's member body in the country of the requester.

ISO copyright office
Case postale 56 • CH-1211 Geneva 20
Tel. + 41 22 749 01 11
Fax + 41 22 734 10 79
E-mail copyright@iso.ch
Web www.iso.ch

Printed in Switzerland

Contents

Foreword	xi
Introduction	xiv
1. Scope	1
2. Normative references	2
3. Terms, definitions, and symbols	3
4. Conformance	7
5. Environment	9
5.1 Conceptual models	9
5.1.1 Translation environment	9
5.1.2 Execution environments	11
5.2 Environmental considerations	17
5.2.1 Character sets	17
5.2.2 Character display semantics	19
5.2.3 Signals and interrupts	20
5.2.4 Environmental limits	20
6. Language	29
6.1 Notation	29
6.2 Concepts	29
6.2.1 Scopes of identifiers	29
6.2.2 Linkages of identifiers	30
6.2.3 Name spaces of identifiers	31
6.2.4 Storage durations of objects	32
6.2.5 Types	33
6.2.6 Representations of types	37
6.2.7 Compatible type and composite type	40
6.3 Conversions	42
6.3.1 Arithmetic operands	42
6.3.2 Other operands	46
6.4 Lexical elements	49
6.4.1 Keywords	50
6.4.2 Identifiers	51
6.4.3 Universal character names	53
6.4.4 Constants	54
6.4.5 String literals	62
6.4.6 Punctuators	63
6.4.7 Header names	64
6.4.8 Preprocessing numbers	65
6.4.9 Comments	66
6.5 Expressions	67

6.5.1	Primary expressions	69
6.5.2	Postfix operators	69
6.5.3	Unary operators	78
6.5.4	Cast operators	81
6.5.5	Multiplicative operators	82
6.5.6	Additive operators	82
6.5.7	Bitwise shift operators	84
6.5.8	Relational operators	85
6.5.9	Equality operators	86
6.5.10	Bitwise AND operator	87
6.5.11	Bitwise exclusive OR operator	88
6.5.12	Bitwise inclusive OR operator	88
6.5.13	Logical AND operator	89
6.5.14	Logical OR operator	89
6.5.15	Conditional operator	90
6.5.16	Assignment operators	91
6.5.17	Comma operator	94
6.6	Constant expressions	95
6.7	Declarations ITeH STANDARD PREVIEW	97
6.7.1	Storage-class specifiers	98
6.7.2	Type specifiers (standards.iteh.ai)	99
6.7.3	Type qualifiers	108
6.7.4	Function specifiers ISO/IEC 9899:1999	112
6.7.5	Declarators https://standards.iteh.ai/catalog/standards/sist/7451fd1d-bac9-4e45-b69f-72f83d536ae1/iso-iec-9899-1999	114
6.7.6	Type names	122
6.7.7	Type definitions	123
6.7.8	Initialization	125
6.8	Statements and blocks	131
6.8.1	Labeled statements	131
6.8.2	Compound statement	132
6.8.3	Expression and null statements	132
6.8.4	Selection statements	133
6.8.5	Iteration statements	135
6.8.6	Jump statements	136
6.9	External definitions	140
6.9.1	Function definitions	141
6.9.2	External object definitions	143
6.10	Preprocessing directives	145
6.10.1	Conditional inclusion	147
6.10.2	Source file inclusion	149
6.10.3	Macro replacement	151
6.10.4	Line control	158
6.10.5	Error directive	159
6.10.6	Pragma directive	159

6.10.7	Null directive	160
6.10.8	Predefined macro names	160
6.10.9	Pragma operator	161
6.11	Future language directions	163
6.11.1	Floating types	163
6.11.2	Linkages of identifiers	163
6.11.3	External names	163
6.11.4	Character escape sequences	163
6.11.5	Storage-class specifiers	163
6.11.6	Function declarators	163
6.11.7	Function definitions	163
6.11.8	Pragma directives	163
6.11.9	Predefined macro names	163
7.	Library	164
7.1	Introduction	164
7.1.1	Definitions of terms	164
7.1.2	Standard headers	165
7.1.3	Reserved identifiers	166
7.1.4	Use of library functions	166
7.2	Diagnostics <assert.h>	169
7.2.1	Program diagnostics	169
7.3	Complex arithmetic <complex.h>	170
7.3.1	Introduction	170
7.3.2	Conventions	171
7.3.3	Branch cuts	171
7.3.4	The CX_LIMITED_RANGE pragma	171
7.3.5	Trigonometric functions	172
7.3.6	Hyperbolic functions	174
7.3.7	Exponential and logarithmic functions	176
7.3.8	Power and absolute-value functions	177
7.3.9	Manipulation functions	178
7.4	Character handling <ctype.h>	181
7.4.1	Character classification functions	181
7.4.2	Character case mapping functions	184
7.5	Errors <errno.h>	186
7.6	Floating-point environment <fenv.h>	187
7.6.1	The FENV_ACCESS pragma	189
7.6.2	Floating-point exceptions	190
7.6.3	Rounding	192
7.6.4	Environment	194
7.7	Characteristics of floating types <float.h>	196
7.8	Format conversion of integer types <inttypes.h>	197
7.8.1	Macros for format specifiers	197
7.8.2	Functions for greatest-width integer types	198

7.9	Alternative spellings <iso646.h>	201
7.10	Sizes of integer types <limits.h>	202
7.11	Localization <locale.h>	203
7.11.1	Locale control	204
7.11.2	Numeric formatting convention inquiry	205
7.12	Mathematics <math.h>	211
7.12.1	Treatment of error conditions	213
7.12.2	The FP_CONTRACT pragma	214
7.12.3	Classification macros	215
7.12.4	Trigonometric functions	217
7.12.5	Hyperbolic functions	220
7.12.6	Exponential and logarithmic functions	222
7.12.7	Power and absolute-value functions	227
7.12.8	Error and gamma functions	229
7.12.9	Nearest integer functions	230
7.12.10	Remainder functions	234
7.12.11	Manipulation functions	235
7.12.12	Maximum, minimum, and positive difference functions	237
7.12.13	Floating multiply-add	238
7.12.14	Comparison macros	239
7.13	Nonlocal jumps <setjmp.h>	242
7.13.1	Save calling environment	242
7.13.2	Restore calling environment	243
7.14	Signal handling <signal.h>	245
7.14.1	Specify signal handling	246
7.14.2	Send signal	247
7.15	Variable arguments <stdarg.h>	248
7.15.1	Variable argument list access macros	248
7.16	Boolean type and values <stdbool.h>	252
7.17	Common definitions <stddef.h>	253
7.18	Integer types <stdint.h>	254
7.18.1	Integer types	254
7.18.2	Limits of specified-width integer types	256
7.18.3	Limits of other integer types	258
7.18.4	Macros for integer constants	259
7.19	Input/output <stdio.h>	261
7.19.1	Introduction	261
7.19.2	Streams	263
7.19.3	Files	265
7.19.4	Operations on files	267
7.19.5	File access functions	269
7.19.6	Formatted input/output functions	273
7.19.7	Character input/output functions	294
7.19.8	Direct input/output functions	299

7.19.9	File positioning functions	300
7.19.10	Error-handling functions	303
7.20	General utilities <stdlib.h>	305
7.20.1	Numeric conversion functions	306
7.20.2	Pseudo-random sequence generation functions	311
7.20.3	Memory management functions	312
7.20.4	Communication with the environment	314
7.20.5	Searching and sorting utilities	317
7.20.6	Integer arithmetic functions	319
7.20.7	Multibyte/wide character conversion functions	320
7.20.8	Multibyte/wide string conversion functions	322
7.21	String handling <string.h>	324
7.21.1	String function conventions	324
7.21.2	Copying functions	324
7.21.3	Concatenation functions	326
7.21.4	Comparison functions	327
7.21.5	Search functions	329
7.21.6	Miscellaneous functions	332
7.22	Type-generic math <tgmath.h>	334
7.23	Date and time <time.h>	337
7.23.1	Components of time	337
7.23.2	Time manipulation functions	338
7.23.3	Time conversion functions	340
7.24	Extended multibyte and wide character utilities <wchar.h>	347
7.24.1	Introduction	347
7.24.2	Formatted wide character input/output functions	348
7.24.3	Wide character input/output functions	366
7.24.4	General wide string utilities	370
7.24.5	Wide character time conversion functions	384
7.24.6	Extended multibyte/wide character conversion utilities	385
7.25	Wide character classification and mapping utilities <wctype.h>	392
7.25.1	Introduction	392
7.25.2	Wide character classification utilities	393
7.25.3	Wide character case mapping utilities	398
7.26	Future library directions	400
7.26.1	Complex arithmetic <complex.h>	400
7.26.2	Character handling <ctype.h>	400
7.26.3	Errors <errno.h>	400
7.26.4	Format conversion of integer types <inttypes.h>	400
7.26.5	Localization <locale.h>	400
7.26.6	Signal handling <signal.h>	400
7.26.7	Boolean type and values <stdbool.h>	400
7.26.8	Integer types <stdint.h>	400
7.26.9	Input/output <stdio.h>	401

7.26.10	General utilities <code><stdlib.h></code>	401
7.26.11	String handling <code><string.h></code>	401
7.26.12	Extended multibyte and wide character utilities <code><wchar.h></code>	401
7.26.13	Wide character classification and mapping utilities <code><wctype.h></code>	401
Annex A	(informative) Language syntax summary	402
A.1	Lexical grammar	402
A.2	Phrase structure grammar	408
A.3	Preprocessing directives	415
Annex B	(informative) Library summary	417
B.1	Diagnostics <code><assert.h></code>	417
B.2	Complex <code><complex.h></code>	417
B.3	Character handling <code><ctype.h></code>	419
B.4	Errors <code><errno.h></code>	419
B.5	Floating-point environment <code><fenv.h></code>	419
B.6	Characteristics of floating types <code><float.h></code>	420
B.7	Format conversion of integer types <code><inttypes.h></code>	420
B.8	Alternative spellings <code><iso646.h></code>	421
B.9	Sizes of integer types <code><limits.h></code>	421
B.10	Localization <code><locale.h></code>	421
B.11	Mathematics <code><math.h></code>	421
B.12	Nonlocal jumps <code><setjmp.h></code>	426
B.13	Signal handling <code><signal.h></code>	426
B.14	Variable arguments <code><stdarg.h></code>	426
B.15	Boolean type and values <code><stdbool.h></code>	426
B.16	Common definitions <code><stddef.h></code>	427
B.17	Integer types <code><stdint.h></code>	427
B.18	Input/output <code><stdio.h></code>	427
B.19	General utilities <code><stdlib.h></code>	429
B.20	String handling <code><string.h></code>	431
B.21	Type-generic math <code><tgmath.h></code>	432
B.22	Date and time <code><time.h></code>	432
B.23	Extended multibyte/wide character utilities <code><wchar.h></code>	433
B.24	Wide character classification and mapping utilities <code><wctype.h></code>	435
Annex C	(informative) Sequence points	437
Annex D	(normative) Universal character names for identifiers	438
Annex E	(informative) Implementation limits	440
Annex F	(normative) IEC 60559 floating-point arithmetic	442
F.1	Introduction	442
F.2	Types	442
F.3	Operators and functions	443

F.4	Floating to integer conversion	445
F.5	Binary-decimal conversion	445
F.6	Contracted expressions	446
F.7	Floating-point environment	446
F.8	Optimization	449
F.9	Mathematics <code><math.h></code>	452
Annex G	(informative) IEC 60559-compatible complex arithmetic	465
G.1	Introduction	465
G.2	Types	465
G.3	Conventions	465
G.4	Conversions	466
G.5	Binary operators	466
G.6	Complex arithmetic <code><complex.h></code>	470
G.7	Type-generic math <code><tgmath.h></code>	478
Annex H	(informative) Language independent arithmetic	479
H.1	Introduction	479
H.2	Types	479
H.3	Notification	483
Annex I	(informative) Common warnings	485
Annex J	(informative) Portability issues	487
J.1	Unspecified behavior	487
J.2	Undefined behavior	490
J.3	Implementation-defined behavior	503
J.4	Locale-specific behavior	510
J.5	Common extensions	511
Bibliography		514
Index		517

iTeh STANDARD PREVIEW
(standards.iteh.ai)

ISO/IEC 9899:1999

<https://standards.iteh.ai/catalog/standards/sist/7451fd1d-bac9-4e45-b69f-72f83d536ae1/iso-iec-9899-1999>

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.
- 2 International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 3.
- 3 In the field of information technology, ISO and IEC have established a joint technical committee, ISO/IEC JTC 1. Draft International Standards adopted by the joint technical committee are circulated to national bodies for voting. Publication as an International Standard requires approval by at least 75% of the national bodies casting a vote.
- 4 International Standard ISO/IEC 9899 was prepared by Joint Technical Committee ISO/IEC JTC 1, *Information technology, Subcommittee SC 22, Programming languages, their environments and system software interfaces*. The Working Group responsible for this standard (WG 14) maintains a site on the World Wide Web at <http://www.dkuug.dk/JTC1/SC22/WG14/> containing additional information relevant to this standard such as a Rationale for many of the decisions made during its preparation and a log of Defect Reports and Responses.
- 5 This second edition cancels and replaces the first edition, ISO/IEC 9899:1990, as amended and corrected by ISO/IEC 9899/COR1:1994, ISO/IEC 9899/AMD1:1995, and ISO/IEC 9899/COR2:1996. Major changes from the previous edition include:
 - restricted character set support via digraphs and **<iso646.h>** (originally specified in AMD1)
 - wide character library support in **<wchar.h>** and **<wctype.h>** (originally specified in AMD1)
 - more precise aliasing rules via effective type
 - restricted pointers
 - variable-length arrays
 - flexible array members
 - **static** and type qualifiers in parameter array declarators
 - complex (and imaginary) support in **<complex.h>**
 - type-generic math macros in **<tgmath.h>**

- the **long long int** type and library functions
- increased minimum translation limits
- additional floating-point characteristics in **<float.h>**
- remove implicit **int**
- reliable integer division
- universal character names (**\u** and **\U**)
- extended identifiers
- hexadecimal floating-point constants and **%a** and **%A printf/scanf** conversion specifiers
- compound literals
- designated initializers
- **//** comments
- extended integer types and library functions in **<inttypes.h>** and **<stdint.h>**
- remove implicit function declaration
- preprocessor arithmetic done in **intmax_t/uintmax_t**
- mixed declarations and code
- new block scopes for selection and iteration statements
- integer constant type rules
- integer promotion rules
- macros with a variable number of arguments
- the **vscanf** family of functions in **<stdio.h>** and **<wchar.h>**
- additional math library functions in **<math.h>**
- floating-point environment access in **<fenv.h>**
- IEC 60559 (also known as IEC 559 or IEEE arithmetic) support
- trailing comma allowed in **enum** declaration
- **%lf** conversion specifier allowed in **printf**
- inline functions
- the **snprintf** family of functions in **<stdio.h>**
- boolean type in **<stdbool.h>**
- idempotent type qualifiers
- empty macro arguments

- new struct type compatibility rules (tag compatibility)
 - additional predefined macro names
 - **_Pragma** preprocessing operator
 - standard pragmas
 - **__func__** predefined identifier
 - **VA_COPY** macro
 - additional **strftime** conversion specifiers
 - LIA compatibility annex
 - deprecate **ungetc** at the beginning of a binary file
 - remove deprecation of aliased array parameters
 - conversion of array to pointer not limited to lvalues
 - relaxed constraints on aggregate and union initialization
 - relaxed restrictions on portable header names
 - **return** without expression not permitted in function that returns a value (and vice versa)
- 6 Annexes D and F form a normative part of this standard; annexes A, B, C, E, G, H, I, J, the bibliography, and the index are for information only. In accordance with Part 3 of the ISO/IEC Directives, this foreword, the introduction, notes, footnotes, and examples are also for information only.

- 1 With the introduction of new devices and extended character sets, new features may be added to this International Standard. Subclauses in the language and library clauses warn implementors and programmers of usages which, though valid in themselves, may conflict with future additions.
- 2 Certain features are *obsolescent*, which means that they may be considered for withdrawal in future revisions of this International Standard. 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 [7.26]) is discouraged.
- 3 This International Standard 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 International Standard. References are used to refer to other related subclauses. Recommendations are provided to give advice or guidance to implementors. Annexes provide additional information and summarize the information contained in this International Standard. A bibliography lists documents that were referred to during the preparation of the standard.
- 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*.

- 1 This International Standard 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 International Standard 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;

1) This International Standard 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.