

This is a preview of "INCITS/ISO/IEC 9899-...". Click here to purchase the full version from the ANSI store.

(ISO/IEC 9899-2011, IDT)

American National Standard

*Information technology — Programming
languages — C*

Developed by



Where IT all begins



This is a preview of "INCITS/ISO/IEC 9899-...". [Click here to purchase the full version from the ANSI store.](#)

PDF disclaimer

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.

Adopted by INCITS (InterNational Committee for Information Technology Standards) as an American National Standard.

Date of ANSI Approval: 5/23/2012

Published by American National Standards Institute,
25 West 43rd Street, New York, New York 10036

Copyright 2012 by Information Technology Industry Council
(ITI). All rights reserved.

These materials are subject to copyright claims of International Standardization Organization (ISO), International Electrotechnical Commission (IEC), American National Standards Institute (ANSI), and Information Technology Industry Council (ITI). Not for resale. No part of this publication may be reproduced in any form, including an electronic retrieval system, without the prior written permission of ITI. All requests pertaining to this standard should be submitted to ITI, 1250 Eye Street NW, Washington, DC 20005.

Printed in the United States of America

Contents

Foreword	xiii
Introduction	xvii
1. Scope	1
2. Normative references	2
3. Terms, definitions, and symbols	3
4. Conformance	8
5. Environment	10
5.1 Conceptual models	10
5.1.1 Translation environment	10
5.1.2 Execution environments	12
5.2 Environmental considerations	22
5.2.1 Character sets	22
5.2.2 Character display semantics	24
5.2.3 Signals and interrupts	25
5.2.4 Environmental limits	25
6. Language	35
6.1 Notation	35
6.2 Concepts	35
6.2.1 Scopes of identifiers	35
6.2.2 Linkages of identifiers	36
6.2.3 Name spaces of identifiers	37
6.2.4 Storage durations of objects	38
6.2.5 Types	39
6.2.6 Representations of types	44
6.2.7 Compatible type and composite type	47
6.2.8 Alignment of objects	48
6.3 Conversions	50
6.3.1 Arithmetic operands	50
6.3.2 Other operands	54
6.4 Lexical elements	57
6.4.1 Keywords	58
6.4.2 Identifiers	59
6.4.3 Universal character names	61
6.4.4 Constants	62
6.4.5 String literals	70
6.4.6 Punctuators	72
6.4.7 Header names	73
6.4.8 Preprocessing numbers	74
6.4.9 Comments	75

6.5	Expressions	76
6.5.1	Primary expressions	78
6.5.2	Postfix operators	79
6.5.3	Unary operators	88
6.5.4	Cast operators	91
6.5.5	Multiplicative operators	92
6.5.6	Additive operators	92
6.5.7	Bitwise shift operators	94
6.5.8	Relational operators	95
6.5.9	Equality operators	96
6.5.10	Bitwise AND operator	97
6.5.11	Bitwise exclusive OR operator	98
6.5.12	Bitwise inclusive OR operator	98
6.5.13	Logical AND operator	99
6.5.14	Logical OR operator	99
6.5.15	Conditional operator	100
6.5.16	Assignment operators	101
6.5.17	Comma operator	105
6.6	Constant expressions	106
6.7	Declarations	108
6.7.1	Storage-class specifiers	109
6.7.2	Type specifiers	111
6.7.3	Type qualifiers	121
6.7.4	Function specifiers	125
6.7.5	Alignment specifier	127
6.7.6	Declarators	128
6.7.7	Type names	136
6.7.8	Type definitions	137
6.7.9	Initialization	139
6.7.10	Static assertions	145
6.8	Statements and blocks	146
6.8.1	Labeled statements	146
6.8.2	Compound statement	147
6.8.3	Expression and null statements	147
6.8.4	Selection statements	148
6.8.5	Iteration statements	150
6.8.6	Jump statements	151
6.9	External definitions	155
6.9.1	Function definitions	156
6.9.2	External object definitions	158
6.10	Preprocessing directives	160
6.10.1	Conditional inclusion	162
6.10.2	Source file inclusion	164
6.10.3	Macro replacement	166

6.10.4	Line control	173
6.10.5	Error directive	174
6.10.6	Pragma directive	174
6.10.7	Null directive	175
6.10.8	Predefined macro names	175
6.10.9	Pragma operator	178
6.11	Future language directions	179
6.11.1	Floating types	179
6.11.2	Linkages of identifiers	179
6.11.3	External names	179
6.11.4	Character escape sequences	179
6.11.5	Storage-class specifiers	179
6.11.6	Function declarators	179
6.11.7	Function definitions	179
6.11.8	Pragma directives	179
6.11.9	Predefined macro names	179
7.	Library	180
7.1	Introduction	180
7.1.1	Definitions of terms	180
7.1.2	Standard headers	181
7.1.3	Reserved identifiers	182
7.1.4	Use of library functions	183
7.2	Diagnostics <assert.h>	186
7.2.1	Program diagnostics	186
7.3	Complex arithmetic <complex.h>	188
7.3.1	Introduction	188
7.3.2	Conventions	189
7.3.3	Branch cuts	189
7.3.4	The CX_LIMITED_RANGE pragma	189
7.3.5	Trigonometric functions	190
7.3.6	Hyperbolic functions	192
7.3.7	Exponential and logarithmic functions	194
7.3.8	Power and absolute-value functions	195
7.3.9	Manipulation functions	196
7.4	Character handling <ctype.h>	200
7.4.1	Character classification functions	200
7.4.2	Character case mapping functions	203
7.5	Errors <errno.h>	205
7.6	Floating-point environment <fenv.h>	206
7.6.1	The FENV_ACCESS pragma	208
7.6.2	Floating-point exceptions	209
7.6.3	Rounding	212
7.6.4	Environment	213
7.7	Characteristics of floating types <float.h>	216

7.8	Format conversion of integer types <code><inttypes.h></code>	217
7.8.1	Macros for format specifiers	217
7.8.2	Functions for greatest-width integer types	218
7.9	Alternative spellings <code><iso646.h></code>	221
7.10	Sizes of integer types <code><limits.h></code>	222
7.11	Localization <code><locale.h></code>	223
7.11.1	Locale control	224
7.11.2	Numeric formatting convention inquiry	225
7.12	Mathematics <code><math.h></code>	231
7.12.1	Treatment of error conditions	233
7.12.2	The <code>FP_CONTRACT</code> pragma	235
7.12.3	Classification macros	235
7.12.4	Trigonometric functions	238
7.12.5	Hyperbolic functions	240
7.12.6	Exponential and logarithmic functions	242
7.12.7	Power and absolute-value functions	247
7.12.8	Error and gamma functions	249
7.12.9	Nearest integer functions	251
7.12.10	Remainder functions	254
7.12.11	Manipulation functions	255
7.12.12	Maximum, minimum, and positive difference functions	257
7.12.13	Floating multiply-add	258
7.12.14	Comparison macros	259
7.13	Nonlocal jumps <code><setjmp.h></code>	262
7.13.1	Save calling environment	262
7.13.2	Restore calling environment	263
7.14	Signal handling <code><signal.h></code>	265
7.14.1	Specify signal handling	266
7.14.2	Send signal	267
7.15	Alignment <code><stdalign.h></code>	268
7.16	Variable arguments <code><stdarg.h></code>	269
7.16.1	Variable argument list access macros	269
7.17	Atomics <code><stdatomic.h></code>	273
7.17.1	Introduction	273
7.17.2	Initialization	274
7.17.3	Order and consistency	275
7.17.4	Fences	278
7.17.5	Lock-free property	279
7.17.6	Atomic integer types	280
7.17.7	Operations on atomic types	282
7.17.8	Atomic flag type and operations	285
7.18	Boolean type and values <code><stdbool.h></code>	287
7.19	Common definitions <code><stddef.h></code>	288
7.20	Integer types <code><stdint.h></code>	289

7.20.1	Integer types	289
7.20.2	Limits of specified-width integer types	291
7.20.3	Limits of other integer types	293
7.20.4	Macros for integer constants	294
7.21	Input/output <stdio.h>	296
7.21.1	Introduction	296
7.21.2	Streams	298
7.21.3	Files	300
7.21.4	Operations on files	302
7.21.5	File access functions	304
7.21.6	Formatted input/output functions	309
7.21.7	Character input/output functions	330
7.21.8	Direct input/output functions	335
7.21.9	File positioning functions	336
7.21.10	Error-handling functions	338
7.22	General utilities <stdlib.h>	340
7.22.1	Numeric conversion functions	341
7.22.2	Pseudo-random sequence generation functions	346
7.22.3	Memory management functions	347
7.22.4	Communication with the environment	350
7.22.5	Searching and sorting utilities	354
7.22.6	Integer arithmetic functions	356
7.22.7	Multibyte/wide character conversion functions	357
7.22.8	Multibyte/wide string conversion functions	359
7.23	_Noreturn <stdnoreturn.h>	361
7.24	String handling <string.h>	362
7.24.1	String function conventions	362
7.24.2	Copying functions	362
7.24.3	Concatenation functions	364
7.24.4	Comparison functions	365
7.24.5	Search functions	367
7.24.6	Miscellaneous functions	371
7.25	Type-generic math <tgmath.h>	373
7.26	Threads <threads.h>	376
7.26.1	Introduction	376
7.26.2	Initialization functions	378
7.26.3	Condition variable functions	378
7.26.4	Mutex functions	380
7.26.5	Thread functions	383
7.26.6	Thread-specific storage functions	386
7.27	Date and time <time.h>	388
7.27.1	Components of time	388
7.27.2	Time manipulation functions	389
7.27.3	Time conversion functions	392

7.28	Unicode utilities <uchar.h>	398
7.28.1	Restartable multibyte/wide character conversion functions	398
7.29	Extended multibyte and wide character utilities <wchar.h>	402
7.29.1	Introduction	402
7.29.2	Formatted wide character input/output functions	403
7.29.3	Wide character input/output functions	421
7.29.4	General wide string utilities	426
7.29.4.1	Wide string numeric conversion functions	426
7.29.4.2	Wide string copying functions	430
7.29.4.3	Wide string concatenation functions	432
7.29.4.4	Wide string comparison functions	433
7.29.4.5	Wide string search functions	435
7.29.4.6	Miscellaneous functions	439
7.29.5	Wide character time conversion functions	439
7.29.6	Extended multibyte/wide character conversion utilities	440
7.29.6.1	Single-byte/wide character conversion functions	441
7.29.6.2	Conversion state functions	441
7.29.6.3	Restartable multibyte/wide character conversion functions	442
7.29.6.4	Restartable multibyte/wide string conversion functions	444
7.30	Wide character classification and mapping utilities <wctype.h>	447
7.30.1	Introduction	447
7.30.2	Wide character classification utilities	448
7.30.2.1	Wide character classification functions	448
7.30.2.2	Extensible wide character classification functions	451
7.30.3	Wide character case mapping utilities	453
7.30.3.1	Wide character case mapping functions	453
7.30.3.2	Extensible wide character case mapping functions	453
7.31	Future library directions	455
7.31.1	Complex arithmetic <complex.h>	455
7.31.2	Character handling <ctype.h>	455
7.31.3	Errors <errno.h>	455
7.31.4	Floating-point environment <fenv.h>	455
7.31.5	Format conversion of integer types <inttypes.h>	455
7.31.6	Localization <locale.h>	455
7.31.7	Signal handling <signal.h>	455
7.31.8	Atomics <stdatomic.h>	455
7.31.9	Boolean type and values <stdbool.h>	456
7.31.10	Integer types <stdint.h>	456
7.31.11	Input/output <stdio.h>	456
7.31.12	General utilities <stdlib.h>	456

7.31.13	String handling <string.h>	456
7.31.14	Date and time <time.h>	456
7.31.15	Threads <threads.h>	456
7.31.16	Extended multibyte and wide character utilities <wchar.h>	456
7.31.17	Wide character classification and mapping utilities <wctype.h>	457
Annex A	(informative) Language syntax summary	458
A.1	Lexical grammar	458
A.2	Phrase structure grammar	465
A.3	Preprocessing directives	473
Annex B	(informative) Library summary	475
B.1	Diagnostics <assert.h>	475
B.2	Complex <complex.h>	475
B.3	Character handling <ctype.h>	477
B.4	Errors <errno.h>	477
B.5	Floating-point environment <fenv.h>	477
B.6	Characteristics of floating types <float.h>	478
B.7	Format conversion of integer types <inttypes.h>	478
B.8	Alternative spellings <iso646.h>	479
B.9	Sizes of integer types <limits.h>	479
B.10	Localization <locale.h>	479
B.11	Mathematics <math.h>	479
B.12	Nonlocal jumps <setjmp.h>	484
B.13	Signal handling <signal.h>	484
B.14	Alignment <stdalign.h>	485
B.15	Variable arguments <stdarg.h>	485
B.16	Atomics <stdatomic.h>	485
B.17	Boolean type and values <stdbool.h>	487
B.18	Common definitions <stddef.h>	487
B.19	Integer types <stdint.h>	487
B.20	Input/output <stdio.h>	488
B.21	General utilities <stdlib.h>	491
B.22	_Noreturn <stdnoreturn.h>	493
B.23	String handling <string.h>	493
B.24	Type-generic math <tgmath.h>	495
B.25	Threads <threads.h>	495
B.26	Date and time <time.h>	496
B.27	Unicode utilities <uchar.h>	497
B.28	Extended multibyte/wide character utilities <wchar.h>	497
B.29	Wide character classification and mapping utilities <wctype.h>	502
Annex C	(informative) Sequence points	503

Annex D (normative) Universal character names for identifiers	504
D.1 Ranges of characters allowed	504
D.2 Ranges of characters disallowed initially	504
Annex E (informative) Implementation limits	505
Annex F (normative) IEC 60559 floating-point arithmetic	507
F.1 Introduction	507
F.2 Types	507
F.3 Operators and functions	508
F.4 Floating to integer conversion	510
F.5 Binary-decimal conversion	510
F.6 The return statement	511
F.7 Contracted expressions	511
F.8 Floating-point environment	511
F.9 Optimization	514
F.10 Mathematics <math.h>	517
F.10.1 Trigonometric functions	518
F.10.2 Hyperbolic functions	520
F.10.3 Exponential and logarithmic functions	520
F.10.4 Power and absolute value functions	524
F.10.5 Error and gamma functions	525
F.10.6 Nearest integer functions	526
F.10.7 Remainder functions	528
F.10.8 Manipulation functions	529
F.10.9 Maximum, minimum, and positive difference functions	530
F.10.10 Floating multiply-add	530
F.10.11 Comparison macros	531
Annex G (normative) IEC 60559-compatible complex arithmetic	532
G.1 Introduction	532
G.2 Types	532
G.3 Conventions	532
G.4 Conversions	533
G.4.1 Imaginary types	533
G.4.2 Real and imaginary	533
G.4.3 Imaginary and complex	533
G.5 Binary operators	533
G.5.1 Multiplicative operators	534
G.5.2 Additive operators	537
G.6 Complex arithmetic <complex.h>	537
G.6.1 Trigonometric functions	539
G.6.2 Hyperbolic functions	539
G.6.3 Exponential and logarithmic functions	543
G.6.4 Power and absolute-value functions	544
G.7 Type-generic math <tgmath.h>	545

Annex H (informative) Language independent arithmetic	546
H.1 Introduction	546
H.2 Types	546
H.3 Notification	550
Annex I (informative) Common warnings	552
Annex J (informative) Portability issues	554
J.1 Unspecified behavior	554
J.2 Undefined behavior	557
J.3 Implementation-defined behavior	571
J.4 Locale-specific behavior	578
J.5 Common extensions	579
Annex K (normative) Bounds-checking interfaces	582
K.1 Background	582
K.2 Scope	583
K.3 Library	583
K.3.1 Introduction	583
K.3.1.1 Standard headers	583
K.3.1.2 Reserved identifiers	584
K.3.1.3 Use of errno	584
K.3.1.4 Runtime-constraint violations	584
K.3.2 Errors <errno.h>	585
K.3.3 Common definitions <stddef.h>	585
K.3.4 Integer types <stdint.h>	585
K.3.5 Input/output <stdio.h>	586
K.3.5.1 Operations on files	586
K.3.5.2 File access functions	588
K.3.5.3 Formatted input/output functions	591
K.3.5.4 Character input/output functions	602
K.3.6 General utilities <stdlib.h>	604
K.3.6.1 Runtime-constraint handling	604
K.3.6.2 Communication with the environment	606
K.3.6.3 Searching and sorting utilities	607
K.3.6.4 Multibyte/wide character conversion functions	610
K.3.6.5 Multibyte/wide string conversion functions	611
K.3.7 String handling <string.h>	614
K.3.7.1 Copying functions	614
K.3.7.2 Concatenation functions	617
K.3.7.3 Search functions	620
K.3.7.4 Miscellaneous functions	621
K.3.8 Date and time <time.h>	624
K.3.8.1 Components of time	624
K.3.8.2 Time conversion functions	624

K.3.9	Extended multibyte and wide character utilities	
	<wchar.h>	627
	K.3.9.1	Formatted wide character input/output functions . . . 628
	K.3.9.2	General wide string utilities 639
	K.3.9.3	Extended multibyte/wide character conversion utilities 647
Annex L (normative)	Analyzability	652
L.1	Scope	652
L.2	Definitions	652
L.3	Requirements	653
Bibliography		654
Index		657

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 International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.
- 3 The main task of the joint technical committee is to prepare International Standards. 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 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.
- 5 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*.
- 6 This third edition cancels and replaces the second edition, ISO/IEC 9899:1999, which has been technically revised. It also incorporates the Technical Corrigenda ISO/IEC 9899:1999/Cor 1:2001, ISO/IEC 9899:1999/Cor 2:2004, and ISO/IEC 9899:1999/Cor 3:2007. Major changes from the previous edition include:
 - conditional (optional) features (including some that were previously mandatory)
 - support for multiple threads of execution including an improved memory sequencing model, atomic objects, and thread-local storage (`<stdatomic.h>` and `<threads.h>`)
 - additional floating-point characteristic macros (`<float.h>`)
 - querying and specifying alignment of objects (`<stdalign.h>`, `<stdlib.h>`)
 - Unicode characters and strings (`<uchar.h>`) (originally specified in ISO/IEC TR 19769:2004)
 - type-generic expressions
 - static assertions
 - anonymous structures and unions

- no-return functions
- macros to create complex numbers (`<complex.h>`)
- support for opening files for exclusive access
- removed the `gets` function (`<stdio.h>`)
- added the `aligned_alloc`, `at_quick_exit`, and `quick_exit` functions (`<stdlib.h>`)
- (conditional) support for bounds-checking interfaces (originally specified in ISO/IEC TR 24731-1:2007)
- (conditional) support for analyzability

7 Major changes in the second edition included:

- restricted character set support via digraphs and `<iso646.h>` (originally specified in ISO/IEC 9899:1990/Amd.1:1995)
- wide character library support in `<wchar.h>` and `<wctype.h>` (originally specified in ISO/IEC 9899:1990/Amd.1:1995)
- 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
- extended integer types
- 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
- specified width integer types and corresponding library functions in `<inttypes.h>` and `<stdint.h>`
- remove implicit function declaration
- preprocessor arithmetic done in `intmax_t/uintmax_t`
- mixed declarations and statements
- 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>`
- treatment of error conditions by math library functions (`math_errhandling`)
- 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 structure 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

ISO/IEC 9899:2011 (E)

© ISO/IEC 2011 – All rights reserved

- 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)

Introduction

- 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 [7.31]) 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*.
- 7 The Working Group responsible for this standard (WG 14) maintains a site on the World Wide Web at <http://www.open-std.org/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.

ISO/IEC 9899:2011 (E)

© ISO/IEC 2011 – All rights reserved

This is a preview of "INCITS/ISO/IEC 9899-...". [Click here to purchase the full version from the ANSI store.](#)

PDF disclaimer

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.

Adopted by INCITS (InterNational Committee for Information Technology Standards) as an American National Standard.

Date of ANSI Approval: 5/23/2012

Published by American National Standards Institute,
25 West 43rd Street, New York, New York 10036

Copyright 2012 by Information Technology Industry Council
(ITI). All rights reserved.

These materials are subject to copyright claims of International Standardization Organization (ISO), International Electrotechnical Commission (IEC), American National Standards Institute (ANSI), and Information Technology Industry Council (ITI). Not for resale. No part of this publication may be reproduced in any form, including an electronic retrieval system, without the prior written permission of ITI. All requests pertaining to this standard should be submitted to ITI, 1250 Eye Street NW, Washington, DC 20005.

Printed in the United States of America

Contents

Foreword	xiii
Introduction	xvii
1. Scope	1
2. Normative references	2
3. Terms, definitions, and symbols	3
4. Conformance	8
5. Environment	10
5.1 Conceptual models	10
5.1.1 Translation environment	10
5.1.2 Execution environments	12
5.2 Environmental considerations	22
5.2.1 Character sets	22
5.2.2 Character display semantics	24
5.2.3 Signals and interrupts	25
5.2.4 Environmental limits	25
6. Language	35
6.1 Notation	35
6.2 Concepts	35
6.2.1 Scopes of identifiers	35
6.2.2 Linkages of identifiers	36
6.2.3 Name spaces of identifiers	37
6.2.4 Storage durations of objects	38
6.2.5 Types	39
6.2.6 Representations of types	44
6.2.7 Compatible type and composite type	47
6.2.8 Alignment of objects	48
6.3 Conversions	50
6.3.1 Arithmetic operands	50
6.3.2 Other operands	54
6.4 Lexical elements	57
6.4.1 Keywords	58
6.4.2 Identifiers	59
6.4.3 Universal character names	61
6.4.4 Constants	62
6.4.5 String literals	70
6.4.6 Punctuators	72
6.4.7 Header names	73
6.4.8 Preprocessing numbers	74
6.4.9 Comments	75

6.5	Expressions	76
6.5.1	Primary expressions	78
6.5.2	Postfix operators	79
6.5.3	Unary operators	88
6.5.4	Cast operators	91
6.5.5	Multiplicative operators	92
6.5.6	Additive operators	92
6.5.7	Bitwise shift operators	94
6.5.8	Relational operators	95
6.5.9	Equality operators	96
6.5.10	Bitwise AND operator	97
6.5.11	Bitwise exclusive OR operator	98
6.5.12	Bitwise inclusive OR operator	98
6.5.13	Logical AND operator	99
6.5.14	Logical OR operator	99
6.5.15	Conditional operator	100
6.5.16	Assignment operators	101
6.5.17	Comma operator	105
6.6	Constant expressions	106
6.7	Declarations	108
6.7.1	Storage-class specifiers	109
6.7.2	Type specifiers	111
6.7.3	Type qualifiers	121
6.7.4	Function specifiers	125
6.7.5	Alignment specifier	127
6.7.6	Declarators	128
6.7.7	Type names	136
6.7.8	Type definitions	137
6.7.9	Initialization	139
6.7.10	Static assertions	145
6.8	Statements and blocks	146
6.8.1	Labeled statements	146
6.8.2	Compound statement	147
6.8.3	Expression and null statements	147
6.8.4	Selection statements	148
6.8.5	Iteration statements	150
6.8.6	Jump statements	151
6.9	External definitions	155
6.9.1	Function definitions	156
6.9.2	External object definitions	158
6.10	Preprocessing directives	160
6.10.1	Conditional inclusion	162
6.10.2	Source file inclusion	164
6.10.3	Macro replacement	166

6.10.4	Line control	173
6.10.5	Error directive	174
6.10.6	Pragma directive	174
6.10.7	Null directive	175
6.10.8	Predefined macro names	175
6.10.9	Pragma operator	178
6.11	Future language directions	179
6.11.1	Floating types	179
6.11.2	Linkages of identifiers	179
6.11.3	External names	179
6.11.4	Character escape sequences	179
6.11.5	Storage-class specifiers	179
6.11.6	Function declarators	179
6.11.7	Function definitions	179
6.11.8	Pragma directives	179
6.11.9	Predefined macro names	179
7.	Library	180
7.1	Introduction	180
7.1.1	Definitions of terms	180
7.1.2	Standard headers	181
7.1.3	Reserved identifiers	182
7.1.4	Use of library functions	183
7.2	Diagnostics <assert.h>	186
7.2.1	Program diagnostics	186
7.3	Complex arithmetic <complex.h>	188
7.3.1	Introduction	188
7.3.2	Conventions	189
7.3.3	Branch cuts	189
7.3.4	The CX_LIMITED_RANGE pragma	189
7.3.5	Trigonometric functions	190
7.3.6	Hyperbolic functions	192
7.3.7	Exponential and logarithmic functions	194
7.3.8	Power and absolute-value functions	195
7.3.9	Manipulation functions	196
7.4	Character handling <ctype.h>	200
7.4.1	Character classification functions	200
7.4.2	Character case mapping functions	203
7.5	Errors <errno.h>	205
7.6	Floating-point environment <fenv.h>	206
7.6.1	The FENV_ACCESS pragma	208
7.6.2	Floating-point exceptions	209
7.6.3	Rounding	212
7.6.4	Environment	213
7.7	Characteristics of floating types <float.h>	216

7.8	Format conversion of integer types <code><inttypes.h></code>	217
7.8.1	Macros for format specifiers	217
7.8.2	Functions for greatest-width integer types	218
7.9	Alternative spellings <code><iso646.h></code>	221
7.10	Sizes of integer types <code><limits.h></code>	222
7.11	Localization <code><locale.h></code>	223
7.11.1	Locale control	224
7.11.2	Numeric formatting convention inquiry	225
7.12	Mathematics <code><math.h></code>	231
7.12.1	Treatment of error conditions	233
7.12.2	The <code>FP_CONTRACT</code> pragma	235
7.12.3	Classification macros	235
7.12.4	Trigonometric functions	238
7.12.5	Hyperbolic functions	240
7.12.6	Exponential and logarithmic functions	242
7.12.7	Power and absolute-value functions	247
7.12.8	Error and gamma functions	249
7.12.9	Nearest integer functions	251
7.12.10	Remainder functions	254
7.12.11	Manipulation functions	255
7.12.12	Maximum, minimum, and positive difference functions	257
7.12.13	Floating multiply-add	258
7.12.14	Comparison macros	259
7.13	Nonlocal jumps <code><setjmp.h></code>	262
7.13.1	Save calling environment	262
7.13.2	Restore calling environment	263
7.14	Signal handling <code><signal.h></code>	265
7.14.1	Specify signal handling	266
7.14.2	Send signal	267
7.15	Alignment <code><stdalign.h></code>	268
7.16	Variable arguments <code><stdarg.h></code>	269
7.16.1	Variable argument list access macros	269
7.17	Atomics <code><stdatomic.h></code>	273
7.17.1	Introduction	273
7.17.2	Initialization	274
7.17.3	Order and consistency	275
7.17.4	Fences	278
7.17.5	Lock-free property	279
7.17.6	Atomic integer types	280
7.17.7	Operations on atomic types	282
7.17.8	Atomic flag type and operations	285
7.18	Boolean type and values <code><stdbool.h></code>	287
7.19	Common definitions <code><stddef.h></code>	288
7.20	Integer types <code><stdint.h></code>	289

7.20.1	Integer types	289
7.20.2	Limits of specified-width integer types	291
7.20.3	Limits of other integer types	293
7.20.4	Macros for integer constants	294
7.21	Input/output <stdio.h>	296
7.21.1	Introduction	296
7.21.2	Streams	298
7.21.3	Files	300
7.21.4	Operations on files	302
7.21.5	File access functions	304
7.21.6	Formatted input/output functions	309
7.21.7	Character input/output functions	330
7.21.8	Direct input/output functions	335
7.21.9	File positioning functions	336
7.21.10	Error-handling functions	338
7.22	General utilities <stdlib.h>	340
7.22.1	Numeric conversion functions	341
7.22.2	Pseudo-random sequence generation functions	346
7.22.3	Memory management functions	347
7.22.4	Communication with the environment	350
7.22.5	Searching and sorting utilities	354
7.22.6	Integer arithmetic functions	356
7.22.7	Multibyte/wide character conversion functions	357
7.22.8	Multibyte/wide string conversion functions	359
7.23	_Noreturn <stdnoreturn.h>	361
7.24	String handling <string.h>	362
7.24.1	String function conventions	362
7.24.2	Copying functions	362
7.24.3	Concatenation functions	364
7.24.4	Comparison functions	365
7.24.5	Search functions	367
7.24.6	Miscellaneous functions	371
7.25	Type-generic math <tgmath.h>	373
7.26	Threads <threads.h>	376
7.26.1	Introduction	376
7.26.2	Initialization functions	378
7.26.3	Condition variable functions	378
7.26.4	Mutex functions	380
7.26.5	Thread functions	383
7.26.6	Thread-specific storage functions	386
7.27	Date and time <time.h>	388
7.27.1	Components of time	388
7.27.2	Time manipulation functions	389
7.27.3	Time conversion functions	392

7.28	Unicode utilities <uchar.h>	398
7.28.1	Restartable multibyte/wide character conversion functions	398
7.29	Extended multibyte and wide character utilities <wchar.h>	402
7.29.1	Introduction	402
7.29.2	Formatted wide character input/output functions	403
7.29.3	Wide character input/output functions	421
7.29.4	General wide string utilities	426
7.29.4.1	Wide string numeric conversion functions	426
7.29.4.2	Wide string copying functions	430
7.29.4.3	Wide string concatenation functions	432
7.29.4.4	Wide string comparison functions	433
7.29.4.5	Wide string search functions	435
7.29.4.6	Miscellaneous functions	439
7.29.5	Wide character time conversion functions	439
7.29.6	Extended multibyte/wide character conversion utilities	440
7.29.6.1	Single-byte/wide character conversion functions	441
7.29.6.2	Conversion state functions	441
7.29.6.3	Restartable multibyte/wide character conversion functions	442
7.29.6.4	Restartable multibyte/wide string conversion functions	444
7.30	Wide character classification and mapping utilities <wctype.h>	447
7.30.1	Introduction	447
7.30.2	Wide character classification utilities	448
7.30.2.1	Wide character classification functions	448
7.30.2.2	Extensible wide character classification functions	451
7.30.3	Wide character case mapping utilities	453
7.30.3.1	Wide character case mapping functions	453
7.30.3.2	Extensible wide character case mapping functions	453
7.31	Future library directions	455
7.31.1	Complex arithmetic <complex.h>	455
7.31.2	Character handling <ctype.h>	455
7.31.3	Errors <errno.h>	455
7.31.4	Floating-point environment <fenv.h>	455
7.31.5	Format conversion of integer types <inttypes.h>	455
7.31.6	Localization <locale.h>	455
7.31.7	Signal handling <signal.h>	455
7.31.8	Atomics <stdatomic.h>	455
7.31.9	Boolean type and values <stdbool.h>	456
7.31.10	Integer types <stdint.h>	456
7.31.11	Input/output <stdio.h>	456
7.31.12	General utilities <stdlib.h>	456

7.31.13	String handling <code><string.h></code>	456
7.31.14	Date and time <code><time.h></code>	456
7.31.15	Threads <code><threads.h></code>	456
7.31.16	Extended multibyte and wide character utilities <code><wchar.h></code>	456
7.31.17	Wide character classification and mapping utilities <code><wctype.h></code>	457
Annex A	(informative) Language syntax summary	458
A.1	Lexical grammar	458
A.2	Phrase structure grammar	465
A.3	Preprocessing directives	473
Annex B	(informative) Library summary	475
B.1	Diagnostics <code><assert.h></code>	475
B.2	Complex <code><complex.h></code>	475
B.3	Character handling <code><ctype.h></code>	477
B.4	Errors <code><errno.h></code>	477
B.5	Floating-point environment <code><fenv.h></code>	477
B.6	Characteristics of floating types <code><float.h></code>	478
B.7	Format conversion of integer types <code><inttypes.h></code>	478
B.8	Alternative spellings <code><iso646.h></code>	479
B.9	Sizes of integer types <code><limits.h></code>	479
B.10	Localization <code><locale.h></code>	479
B.11	Mathematics <code><math.h></code>	479
B.12	Nonlocal jumps <code><setjmp.h></code>	484
B.13	Signal handling <code><signal.h></code>	484
B.14	Alignment <code><stdalign.h></code>	485
B.15	Variable arguments <code><stdarg.h></code>	485
B.16	Atomics <code><stdatomic.h></code>	485
B.17	Boolean type and values <code><stdbool.h></code>	487
B.18	Common definitions <code><stddef.h></code>	487
B.19	Integer types <code><stdint.h></code>	487
B.20	Input/output <code><stdio.h></code>	488
B.21	General utilities <code><stdlib.h></code>	491
B.22	<code>_Noreturn</code> <code><stdnoreturn.h></code>	493
B.23	String handling <code><string.h></code>	493
B.24	Type-generic math <code><tgmath.h></code>	495
B.25	Threads <code><threads.h></code>	495
B.26	Date and time <code><time.h></code>	496
B.27	Unicode utilities <code><uchar.h></code>	497
B.28	Extended multibyte/wide character utilities <code><wchar.h></code>	497
B.29	Wide character classification and mapping utilities <code><wctype.h></code>	502
Annex C	(informative) Sequence points	503

Annex D (normative) Universal character names for identifiers	504
D.1 Ranges of characters allowed	504
D.2 Ranges of characters disallowed initially	504
Annex E (informative) Implementation limits	505
Annex F (normative) IEC 60559 floating-point arithmetic	507
F.1 Introduction	507
F.2 Types	507
F.3 Operators and functions	508
F.4 Floating to integer conversion	510
F.5 Binary-decimal conversion	510
F.6 The return statement	511
F.7 Contracted expressions	511
F.8 Floating-point environment	511
F.9 Optimization	514
F.10 Mathematics <math.h>	517
F.10.1 Trigonometric functions	518
F.10.2 Hyperbolic functions	520
F.10.3 Exponential and logarithmic functions	520
F.10.4 Power and absolute value functions	524
F.10.5 Error and gamma functions	525
F.10.6 Nearest integer functions	526
F.10.7 Remainder functions	528
F.10.8 Manipulation functions	529
F.10.9 Maximum, minimum, and positive difference functions	530
F.10.10 Floating multiply-add	530
F.10.11 Comparison macros	531
Annex G (normative) IEC 60559-compatible complex arithmetic	532
G.1 Introduction	532
G.2 Types	532
G.3 Conventions	532
G.4 Conversions	533
G.4.1 Imaginary types	533
G.4.2 Real and imaginary	533
G.4.3 Imaginary and complex	533
G.5 Binary operators	533
G.5.1 Multiplicative operators	534
G.5.2 Additive operators	537
G.6 Complex arithmetic <complex.h>	537
G.6.1 Trigonometric functions	539
G.6.2 Hyperbolic functions	539
G.6.3 Exponential and logarithmic functions	543
G.6.4 Power and absolute-value functions	544
G.7 Type-generic math <tgmath.h>	545

Annex H (informative) Language independent arithmetic	546
H.1 Introduction	546
H.2 Types	546
H.3 Notification	550
Annex I (informative) Common warnings	552
Annex J (informative) Portability issues	554
J.1 Unspecified behavior	554
J.2 Undefined behavior	557
J.3 Implementation-defined behavior	571
J.4 Locale-specific behavior	578
J.5 Common extensions	579
Annex K (normative) Bounds-checking interfaces	582
K.1 Background	582
K.2 Scope	583
K.3 Library	583
K.3.1 Introduction	583
K.3.1.1 Standard headers	583
K.3.1.2 Reserved identifiers	584
K.3.1.3 Use of errno	584
K.3.1.4 Runtime-constraint violations	584
K.3.2 Errors <errno.h>	585
K.3.3 Common definitions <stddef.h>	585
K.3.4 Integer types <stdint.h>	585
K.3.5 Input/output <stdio.h>	586
K.3.5.1 Operations on files	586
K.3.5.2 File access functions	588
K.3.5.3 Formatted input/output functions	591
K.3.5.4 Character input/output functions	602
K.3.6 General utilities <stdlib.h>	604
K.3.6.1 Runtime-constraint handling	604
K.3.6.2 Communication with the environment	606
K.3.6.3 Searching and sorting utilities	607
K.3.6.4 Multibyte/wide character conversion functions	610
K.3.6.5 Multibyte/wide string conversion functions	611
K.3.7 String handling <string.h>	614
K.3.7.1 Copying functions	614
K.3.7.2 Concatenation functions	617
K.3.7.3 Search functions	620
K.3.7.4 Miscellaneous functions	621
K.3.8 Date and time <time.h>	624
K.3.8.1 Components of time	624
K.3.8.2 Time conversion functions	624

K.3.9	Extended multibyte and wide character utilities	
	<wchar.h>	627
K.3.9.1	Formatted wide character input/output functions	628
K.3.9.2	General wide string utilities	639
K.3.9.3	Extended multibyte/wide character conversion utilities	647
Annex L (normative)	Analyzability	652
L.1	Scope	652
L.2	Definitions	652
L.3	Requirements	653
Bibliography		654
Index		657

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 International Standards are drafted in accordance with the rules given in the ISO/IEC Directives, Part 2.
- 3 The main task of the joint technical committee is to prepare International Standards. 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 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.
- 5 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*.
- 6 This third edition cancels and replaces the second edition, ISO/IEC 9899:1999, which has been technically revised. It also incorporates the Technical Corrigenda ISO/IEC 9899:1999/Cor 1:2001, ISO/IEC 9899:1999/Cor 2:2004, and ISO/IEC 9899:1999/Cor 3:2007. Major changes from the previous edition include:
 - conditional (optional) features (including some that were previously mandatory)
 - support for multiple threads of execution including an improved memory sequencing model, atomic objects, and thread-local storage (`<stdatomic.h>` and `<threads.h>`)
 - additional floating-point characteristic macros (`<float.h>`)
 - querying and specifying alignment of objects (`<stdalign.h>`, `<stdlib.h>`)
 - Unicode characters and strings (`<uchar.h>`) (originally specified in ISO/IEC TR 19769:2004)
 - type-generic expressions
 - static assertions
 - anonymous structures and unions

- no-return functions
- macros to create complex numbers (`<complex.h>`)
- support for opening files for exclusive access
- removed the `gets` function (`<stdio.h>`)
- added the `aligned_alloc`, `at_quick_exit`, and `quick_exit` functions (`<stdlib.h>`)
- (conditional) support for bounds-checking interfaces (originally specified in ISO/IEC TR 24731-1:2007)
- (conditional) support for analyzability

7 Major changes in the second edition included:

- restricted character set support via digraphs and `<iso646.h>` (originally specified in ISO/IEC 9899:1990/Amd.1:1995)
- wide character library support in `<wchar.h>` and `<wctype.h>` (originally specified in ISO/IEC 9899:1990/Amd.1:1995)
- 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
- extended integer types
- 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
- specified width integer types and corresponding library functions in `<inttypes.h>` and `<stdint.h>`
- remove implicit function declaration
- preprocessor arithmetic done in `intmax_t/uintmax_t`
- mixed declarations and statements
- 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>`
- treatment of error conditions by math library functions (`math_errhandling`)
- 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 structure 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

ISO/IEC 9899:2011 (E)

© ISO/IEC 2011 – All rights reserved

- 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)

Introduction

- 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 [7.31]) 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*.
- 7 The Working Group responsible for this standard (WG 14) maintains a site on the World Wide Web at <http://www.open-std.org/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.

ISO/IEC 9899:2011 (E)

© ISO/IEC 2011 – All rights reserved