

C^{Fourth Edition} The Complete Reference

Covers C99, the New
ANSI/ISO Standard for C

The Ultimate Resource on C—Thoroughly Updated for the new Standard

A new ANSI/ISO standard for C, called C99, has been recently adopted and Herb Schildt, the world's leading programming author, has updated and expanded his best-selling reference on C to cover it. Whether you are a beginning C programmer or a seasoned pro, the answers to all your C questions can be found in this one-stop resource. In this authoritative guide, Schildt details the C language, its libraries, and

applications, providing insider tips hundreds of examples, and expertly crafted explanations. As a special bonus, the book concludes by developing a C interpreter, which you can use as-is or expand on your own! And just as you'd expect, everything is presented in the clear, concise, uncompromising style that has made Herb the choice of millions.

Inside you'll find:

- Comprehensive coverage of the C language, including both C89 (the original version of C) and the new features added by C99
- Detailed explanations of each keyword, data type, and operator
- In-depth discussions of pointer, disk I/O, and dynamic allocation
- Complete descriptions of the entire C function library
- Descriptions of the new features added by C99, such as restricted pointers, boolean and complex data types, the inline keyword, variable-length arrays, and the long long data types
- Real-world algorithms and applications, such as stacks, queues, trees, sparse arrays, and sorting. You'll even learn about AI-based searching techniques
- Tips on working efficiently in the C programming environment
- Advice on porting and debugging
- Complete source code for a C Interpreter that you can use as-is or enhance to fit your needs

Herbert Schildt is a leading authority on C and was member of the ANSI/ISO committee that standardized C. He is the author of *Teach Yourself C*, *C++ The Complete Reference*, *Windows 2000 Programming from the ground Up*, *Java: The Complete Reference*, and many other best-sellers

Mc
Graw
Hill
Education

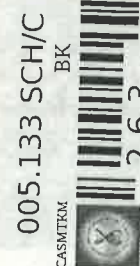
Professional



Covers C99, the New
ANSI/ISO Standard for C

The
Complete
Reference

C
Fourth
Edition



Schildt

Mc
Graw
Hill
Education

INDIAN EDITION

Covers C99, the New
ANSI/ISO Standard for C

For sale in
India, Pakistan,
Nepal, Bangladesh,
Sri Lanka and Bhutan
only

For all
versions
of C

The Complete Reference

C Fourth Edition

"Herb Schildt tells his programmers what they want to know—simply, clearly, concisely, and authoritatively."

—ACM Computing Reviews

In-depth coverage of the
C language and function
libraries

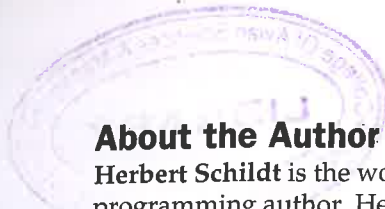


Cover all the newest
C features, including
restricted pointers, inline
function, variable-length
arrays, and complex math

Includes hundreds of
examples and
sample
applications

Herbert Schildt

Best-selling Programming Author with
More Than 2.5 Million Books Sold!



About the Author

Herbert Schildt is the world's leading programming author. He is an authority on the C and C++ languages, a master Windows programmer, and an expert on Java. His programming books have sold more than 2.5 million copies worldwide and have been translated into all major foreign languages. He is the author of numerous bestsellers, including *C++: The Complete Reference*, *Teach Yourself C*, *Teach Yourself C++*, *C++ from the Ground Up*, *Windows 2000 Programming from the Ground Up*, and *Java: The Complete Reference*. Schildt holds a master's degree in computer science from the University of Illinois. He can be reached at his consulting office at (217) 586-4683.



C: The Complete Reference, Fourth Edition

Herbert Schildt

**Mc
Graw
Hill
Education**

McGraw Hill Education (India) Private Limited

NEW DELHI

McGraw Hill Education Offices

New Delhi New York St Louis San Francisco Auckland Bogotá Caracas
Kuala Lumpur Lisbon London Madrid Mexico City Milan Montreal
San Juan Santiago Singapore Sydney Tokyo Toronto



McGraw Hill Education (India) Private Limited

C: The Complete Reference, Fourth Edition

Copyright 2000 by The McGraw-Hill Companies, Inc.

All rights reserved. No part of this publication may be reproduced or distributed in any form or by any means, or stored in a database or retrieval system without the prior written permission of the publisher

McGraw Hill Education (India) Edition 2000

35th reprint 2013

RACDRQODRDQA

Reprinted in India by arrangement with The McGraw-Hill Companies, Inc., New York

Sales territories: India, Pakistan, Nepal, Bangladesh, Sri Lanka and Bhutan

ISBN-13: 978-0-07-041183-8

ISBN-10: 0-07-041183-2

Published by McGraw Hill Education (India) Private Limited,
P-24, Green Park Extension, New Delhi 110 016, and printed at
Akash Press, New Delhi 110 020

Contents

Preface xxiii

PART I

Foundational C

1	An Overview of C	3
	A Brief History of C	4
	C Is a Middle-Level Language	5
	C Is a Structured Language	6
	C Is a Programmer's Language	8
	Compilers vs. Interpreters	9
	The Form of a C Program	10
	The Library and Linking	11
	Separate Compilation	12
	Compiling a C Program	13
	C's Memory Map	13
	C vs. C++	14
	Review of Terms	15

2	Expressions	17
	The Basic Data Types	18
	Modifying the Basic Types	18
	Identifier Names	20
	Variables	21
	Where Variables Are Declared	21
	Local Variables	22
	Formal Parameters	25
	Global Variables	26
	The Four C Scopes	27
	Type Qualifiers	28
	const	28
	volatile	30
	Storage Class Specifiers	30
	extern	31
	static Variables	33
	register Variables	35
	Variable Initializations	36
	Constants	37
	Hexadecimal and Octal Constants	38
	String Constants	38
	Backslash Character Constants	39
	Operators	40
	The Assignment Operator	40
	Arithmetic Operators	43
	The Increment and Decrement Operators	44
	Relational and Logical Operators	46
	Bitwise Operators	48
	The ? Operator	53
	The & and * Pointer Operators	53
	The Compile-Time Operator sizeof	55
	The Comma Operator	56
	The Dot (.) and Arrow (->) Operators	56
	The [] and () Operators	57
	Precedence Summary	58
	Expressions	58
	Order of Evaluation	58
	Type Conversion in Expressions	59
	Casts	60
	Spacing and Parentheses	61
3	Statements	63
	True and False in C	64

	Selection Statements	64
	if	64
	Nested ifs	66
	The if-else-if Ladder	67
	The ? Alternative	69
	The Conditional Expression	72
	switch	72
	Nested switch Statements	75
	Iteration Statements	76
	The for Loop	76
	for Loop Variations	77
	The Infinite Loop	82
	for Loops with No Bodies	82
	Declaring Variables Within a for Loop	83
	The while Loop	83
	The do-while Loop	86
	Jump Statements	87
	The return Statement	87
	The goto Statement	88
	The break Statement	89
	The exit() Function	90
	The continue Statement	91
	Expression Statements	93
	Block Statements	93
4	Arrays and Strings	95
	Single-Dimension Arrays	96
	Generating a Pointer to an Array	97
	Passing Single-Dimension Arrays to Functions	98
	Strings	99
	Two-Dimensional Arrays	101
	Arrays of Strings	106
	Multidimensional Arrays	107
	Indexing Pointers	108
	Array Initialization	110
	Unsize Array Initializations	112
	Variable-Length Arrays	113
	A Tic-Tac-Toe Example	114
5	Pointers	119
	What Are Pointers?	120
	Pointer Variables	121
	The Pointer Operators	121

Non-White-Space Characters in the Control String	226
You Must Pass scanf() Addresses	226
Format Modifiers	226
Suppressing Input	227
9 File I/O	229
C vs. C++ File I/O	230
Standard C vs. Unix File I/O	230
Streams and Files	230
Streams	230
Files	231
File System Basics	231
The File Pointer	232
Opening a File	232
Closing a File	232
Writing a Character	235
Reading a Character	235
Using fopen(), getc(), putc(), and fclose()	236
Using feof()	236
Working with Strings: fputs() and fgets()	238
rewind()	239
ferror()	240
Erasing Files	241
Flushing a Stream	243
fread() and fwrite()	244
Using fread() and fwrite()	245
fseek() and Random-Access I/O	245
fprintf() and fscanf()	253
The Standard Streams	254
The Console I/O Connection	256
Using freopen() to Redirect the Standard Streams	257
10 The Preprocessor and Comments	258
The Preprocessor	261
#define	262
Defining Function-like Macros	262
#error	264
#include	265
Conditional Compilation Directives	265
#if, #else, #elif, and #endif	266
#ifdef and #ifndef	266
#undef	269
Using defined	270
#line	270
	271

#pragma	272
The # and ## Preprocessor Operators	272
Predefined Macro Names	273
Comments	274
Single Line Comments	275

PART II

The C99 Standard

11 C99	279
C89 vs. C99: An Overview	280
Features Added	280
Features Removed	281
Features Changed	281
restrict-Qualified Pointers	282
inline	282
New Built-in Data Types	284
_Bool	284
_Complex and _Imaginary	284
The long long Integer Types	285
Array Enhancements	285
Variable-Length Arrays	285
Use of Type Qualifiers in an Array Declaration	286
Single-Line Comments	286
Interspersed Code and Declarations	286
Preprocessor Changes	287
Variable Argument Lists	287
The _Pragma Operator	288
Built-in Pragmas	288
Additional Built-in Macros	289
Declaring Variables Within a for Loop	289
Compound Literals	290
Flexible Array Structure Members	291
Designated Initializers	291
Additions to the printf() and scanf() Family of Functions	292
New Libraries in C99	293
The __func__ Predefined Identifier	293
Increased Translation Limits	294
Implicit int No Longer Supported	294
Implicit Function Declarations Have Been Removed	296
Restrictions on return	296
Extended Integer Types	297
Changes to the Integer Promotion Rules	297

PART III

The C Standard Library

12	Linking, Libraries, and Headers	301
	The Linker	302
	Separate Compilation	302
	Relocatable vs. Absolute Code	303
	Linking with Overlays	303
	Linking with DLLs	304
	The C Standard Library	305
	Library Files vs. Object Files	305
	Headers	305
	Macros in Headers	307
	Redefinition of Library Functions	308
13	I/O Functions	309
	clearerr	310
	fclose	312
	feof	313
	ferror	313
	fflush	314
	fgetc	315
	fgetpos	316
	fgets	317
	fopen	318
	fprintf	320
	fputc	321
	fputs	321
	fread	322
	freopen	323
	fscanf	324
	fseek	325
	fsetpos	326
	ftell	327
	fwrite	328
	getc	329
	getchar	330
	gets	331
	perror	332
	printf	332
	Format Modifiers for printf() Added by C99	335
	putc	336
	putchar	337
	puts	337

remove	338
rename	339
rewind	340
scanf	340
Format Modifiers for scanf() Added by C99	344
setbuf	345
setvbuf	345
snprintf	346
sprintf	347
sscanf	347
tmpfile	348
tmpnam	349
ungetc	350
vprintf, fprintf, vsprintf, and vsnprintf	351
vscanf, vfscanf, and vsscanf	352
14 String and Character Functions	353
isalnum	354
isalpha	355
isblank	356
iscntrl	357
isdigit	358
isgraph	358
islower	359
isprint	360
ispunct	361
isspace	362
isupper	362
isxdigit	363
memchr	364
memcmp	365
memcpy	366
memmove	367
memset	368
strcat	368
strchr	369
strcmp	370
strcoll	371
strcpy	372
strcspn	372
strerror	373
strlen	373
strncat	374
strncmp	375
strncpy	376

18	Utility Functions	439
	abort	440
	abs	441
	assert	441
	atexit	442
	atof	443
	atoi	444
	atol	445
	atoll	446
	bsearch	446
	div	448
	exit	449
	_Exit	450
	getenv	450
	labs	451
	llabs	451
	ldiv	452
	lldiv	453
	longjmp	453
	mblen	454
	mbstowcs	455
	mbtowc	456
	qsort	456
	raise	458
	rand	459
	setjmp	459
	signal	460
	srand	460
	strtod	461
	strtof	463
	strtol	463
	strtold	464
	strtoll	465
	strtoul	465
	strtoull	466
	system	467
	va_arg, va_copy, va_start, and va_end	467
	wcstombs	469
	wctomb	470
19	Wide-Character Functions	471
	Wide-Character Classification Functions	472
	Wide-Character I/O Functions	474
	Wide-Character String Functions	477

Wide-Character String Conversion Functions	478
Wide-Character Array Functions	479
Multibyte/Wide-Character Conversion Functions	480

20	Library Features Added by C99	483
	The Complex Library	484
	The Floating-Point Environment Library	488
	The <stdint.h> Header	488
	Integer Format Conversion Functions	490
	Type-Generic Math Macros	490
	The <stdbool.h> Header	493

Part VI

Algorithms and Applications

21	Sorting and Searching	497
	Sorting	498
	Classes of Sorting Algorithms	498
	Judging Sorting Algorithms	499
	The Bubble Sort	500
	Sorting by Selection	504
	Sorting by Insertion	505
	Improved Sorts	506
	The Shell Sort	506
	The Quicksort	508
	Choosing a Sort	511
	Sorting Other Data Structures	511
	Sorting Strings	512
	Sorting Structures	513
	Sorting Random-Access Disk Files	515
	Searching	518
	Searching Methods	519
	The Sequential Search	519
	The Binary Search	519
22	Queues, Stacks, Linked Lists, and Trees	521
	Queues	522
	The Circular Queue	528
	Stacks	531
	Linked Lists	536
	Singly Linked Lists	536
	Doubly Linked Lists	541
	A Mailing List Example	546
	Binary Trees	553

23	Sparse Arrays	
	Understanding the Need for Sparse Arrays	563
	The Linked-List Sparse Array	564
	Analysis of the Linked-List Approach	565
	The Binary-Tree Approach to Sparse Arrays	568
	Analysis of the Binary-Tree Approach	569
	The Pointer-Array Approach to Sparse Arrays	571
	Analysis of the Pointer-Array Approach	572
	Hashing	575
	Analysis of Hashing	575
	Choosing an Approach	579
		580
24	Expression Parsing and Evaluation	
	Expressions	581
	Dissecting an Expression	582
	Expression Parsing	584
	A Simple Expression Parser	586
	Adding Variables to the Parser	588
	Syntax Checking in a Recursive-Descent Parser	595
		604
25	AI-Based Problem Solving	
	Representation and Terminology	605
	Combinatorial Explosions	606
	Search Techniques	608
	Evaluating a Search	610
	A Graphic Representation	610
	The Depth-First Search	611
	Analysis of the Depth-First Search	613
	The Breadth-First Search	624
	Analysis of the Breadth-First Search	625
	Adding Heuristics	626
	The Hill-Climbing Search	626
	Analysis of Hill Climbing	628
	The Least-Cost Search	635
	Analysis of the Least-Cost Search	635
	Choosing a Search Technique	636
	Finding Multiple Solutions	636
	Path Removal	637
	Node Removal	638
	Finding the "Optimal" Solution	639
	Back to the Lost Keys	645
		652

Part V

Software Development Using C

26	Building a Windows 2000 Skeleton	659
	Windows 2000 Programming Perspective	660
	The Desktop Model	661
	The Mouse	661
	Icons, Bitmaps, and Graphics	661
	Menus, Controls, and Dialog Boxes	661
	The Win32 Application Programming Interface	662
	Components of a Window	662
	How Windows and Your Program Interact	663
	Some Windows 2000 Application Basics	664
	WinMain()	664
	The Window Procedure	664
	Window Classes	665
	The Message Loop	665
	Windows Data Types	665
	A Windows 2000 Skeleton	666
	Defining the Window Class	669
	Creating a Window	672
	The Message Loop	674
	The Window Function	675
	Definition File No Longer Needed	676
	Naming Conventions	676
27	Software Engineering Using C	679
	Top-Down Design	680
	Outlining Your Program	680
	Choosing a Data Structure	682
	Bulletproof Functions	682
	Using MAKE	685
	Using Macros in MAKE	689
	Using an Integrated Development Environment	689
28	Efficiency, Porting, and Debugging	691
	Efficiency	692
	The Increment and Decrement Operators	692
	Using Register Variables	693
	Pointers vs. Array Indexing	694
	Use of Functions	694

Porting Programs	698
Using #define	698
Operating-System Dependencies	699
Differences in Data Sizes	699
Debugging	700
Order-of-Evaluation Errors	700
Pointer Problems	701
Interpreting Syntax Errors	703
One-Off Errors	704
Boundary Errors	705
Function Prototype Omissions	706
Argument Errors	708
Stack Overruns	708
Using a Debugger	708
Debugging Theory in General	709

Part IV

A C Interpreter

29 A C Interpreter	713
The Practical Importance of Interpreters	714
The Little C Specifications	715
Some Little C Restrictions	716
Interpreting a Structured Language	718
An Informal Theory of C	719
C Expressions	720
Evaluating Expressions	721
The Expression Parser	722
Reducing the Source Code to Its Components	723
The Little C Recursive-Descent Parser	730
The Little C Interpreter	744
The Interpreter Prescan	745
The main() Function	748
The interp_block() Function	749
Handling Local Variables	766
Calling User-Defined Functions	767
Assigning Values to Variables	771
Executing an if Statement	772
Processing a while Loop	773
Processing a do-while Loop	774
The for Loop	775
The Little C Library Functions	776
Compiling and Linking the Little C Interpreter	780

Demonstrating Little C	780
Improving Little C	785
Expanding Little C	786
Adding New C Features	786
Adding Ancillary Features	787
Index	789

Index

- & (bitwise operator), 49, 50
- & (pointer operator), 53-55, 121, 186, 189
- &&, 46, 47, 48
- < >, 265, 266
- *->, 56-57, 189-190, 192, 194, 197
- * (multiplication operator), 43
- * (pointer operator), 53-55, 121, 131, 189
- * (printf() placeholder), 220-221
- |, 49, 50
- ||, 46, 47, 48
- [], 57, 96
- ^, 49, 50, 225
- ;, 53
- , (comma operator), 56
- { }, 8, 22, 28, 64, 88, 93, 111, 290, 719
- . (dot operator), 56-57, 176-177, 192, 194, 197
- !, 46, 47, 48
- !=, 47, 48
- =, 36, 40
- ==, 47, 48
- <, 47, 48
- <<, 49, 51-52
- <=, 47, 48
- , 43, 44
- , 44-45, 692-693, 700
- (), 57, 61, 135
- % (format specifier), 213, 221
- % (modulus operator), 43
- +, 43
- ++, 44-45, 692-693, 700
- # (preprocessor directive), 262

(preprocessor operator), 272-273
 # (printf() modifier), 220
 ## (preprocessor operator), 272-273
 ?, 53, 69-71, 72
 >, 47, 48
 >>, 49, 51-52
 >=, 47, 48
 ; (semicolon), 76, 93, 174
 /, 43
 /* */, 274-275
 //, 275
 ~, 49, 52-53

A

abort(), 440-441
 abs(), 441
 acos(), 386-387
 acosh(), 387
 Ada, 5, 6
 Address, memory
 & operator used to return, 54
 displaying, 215
 inputting, 224
 pointer as, 53
 Addressing, 32-bit versus 16-bit, 662
 ALGOL, 6
 AND
 & bitwise operator, 49, 50
 && logical operator, 47
 ANSI C standard, 4
 API (Application Program Interface), 304, 662
 argc, 155-158
 Arguments, function, 149-155
 call by reference passing
 convention, 150-151
 call by value passing
 convention, 149-150

command line, 155-158
 passing arrays as, 152-155
 argv, 130, 155-158
 Arithmetic operators, 43-46
 precedence of, 45
 Array(s)
 accessing with pointers, 109,
 128-129
 bounds checking on, 6, 97, 354,
 705-706
 declaration, using type
 qualifiers in, 286
 definition of, 96
 designated initializers and,
 291-292
 dynamically allocated, 140-142
 generating pointer to, 97-98
 indexing versus pointer
 arithmetic, 694
 initialization, 110-113
 logical versus physical, 565
 multidimensional, 107-108
 passing, to functions, 98-99, 103,
 107-108, 152-155
 of pointers, 129-130, 572-575
 single-dimension, 96-97
 sorting, 498-511
 square brackets for indexing, 57
 of strings, 106-107
 of structures, 178-185
 within structures, 192, 291
 two-dimensional, 101-105
 unsized, 112-113
 variable-length, 113-114, 285
 See also Sparse arrays
 Arrow operator (->), 56-57, 189-190,
 192, 194, 197
 Art of C (Schildt), The, 719

Artificial intelligence (AI) and
 problem solving. *See* Problem
 solving, AI-based
 asctime(), 418-419
 asin(), 387-388
 asinh(), 388
 Assembly language, 5, 8
 assert(), 441-442
 <assert.h> header, 442
 Assignment
 compound, 42-43
 multiple, 42
 operator, 40
 pointer, 122-123
 shorthand, 43
 structure, 177
 type conversion in, 40-42
 atan(), 388-389
 atan2(), 390
 atanh(), 389-390
 atexit(), 442-443
 atof(), 443-444
 atoi(), 444-445
 atol(), 445-446
 atoll(), 446
 auto keyword, 22
 Automatic variables, 22

B

B language, 4
 Backslash character constants,
 table of, 39
 BASIC, 5, 7, 8
 BCPL language, 4
 Binary
 code, 9
 search, 519-520
 trees, 553-561, 569-571
 Bit shift operators (>> and <<), 51-52

Bit-fields, 174, 195-198
 Bitmaps, 660
 Bitwise operation, definition of,
 48-49
 Bitwise operators, 48-53
 table of, 49
 Block
 scope, 28, 148
 statements, 93-94
 structured language, 6, 149
 _Bool data type, 18, 46, 65, 284, 493
 bool macro, 284, 493
 __bool_true_false_are_defined
 macro, 493
 break statement, 73, 74, 75, 89-90
 bsearch(), 446-448, 518
 Bubble sort, 500-503

C

C, informal theory of, 719-722
 C versus C++, 14
 C Programming Language
 (Kernighan & Ritchie), The, 4
 C89, 1-2, 4
 C99, 2, 4, 278-297
 C/C++ Annotated Archives
 (Schildt), The, 719
 C++, 1-2, 4, 5, 9
 C++: The Complete Reference
 (Schildt), 14
 Call by reference, 150-152, 189
 Call by value, 149-150
 CALLBACK calling convention, 664
 Callback function, 664
 Calling sequence, 694, 695
 calloc(), 434-435
 cbrt(), 391
 Case sensitivity in C, 11, 21
 case statement, 73-75

definition of, 18, 58
 dissecting, 584-586
 evaluation order, 58-59
 function calls used in, 70, 160
 numeric, definition of, 582
 parsing, 582, 586-588
 production rules, 586, 721
 statements, 93
 type conversion in, 59-60
See also Parser, recursive descent

extern storage class specifier, 30-33
 External names, 21

F

`fabs()`, 396
 false macro, 284, 493
`fclose()`, 161, 235, 312
`fdim()`, 397
`FE_DFL_ENV` macro, 488
`<fenv.h>` header, 293, 488
`FENV_ACCESS` pragma, 488
`fenv_t` type, 488
`feof()`, 238-239, 313
`ferror()`, 235, 236, 241-243, 313-314
`fexcept_t` type, 488
`fflush()`, 244, 314-315
`fgetc()`, 236, 315-316
`fgetpos()`, 316
`fgets()`, 210, 239-240, 256-257, 317-318
 FIFO (first-in, first-out), 522
 File(s)
 in C I/O system, 230, 231-232
 checking for EOF in binary, 238
 closing, 235
 control structure, 232
 erasing, 243-244
 make, 686

opening, 232-235
 pointer, 232, 310
 scope, 27, 149
 sorting random-access disk, 515-518
 FILE data type, 232, 305, 310
 File position indicator, 231
 resetting, 240-241
`_FILE_` predefined macro, 271, 273
 float data type, 18, 19
 Floating-point environment
 functions, table of, 489
 library, 488
`floor()`, 397
`fma()`, 398
`fmax()`, 398
`fmin()`, 398-399
`fmod()`, 399
`fopen()`, 232-235, 318-319
`FOPEN_MAX` macro, 232, 235, 319
 for loop, 6, 76-83, 90, 92
 declaring variables in, 279
 general form of, 76
 infinite, 82
 variations of, 77-82
 with no body, 82-83
 FORTRAN, 5, 7
`fpclassify()`, 384
`fpos_t` data type, 232, 310
`fprintf()`, 254-256, 320
`fputc()`, 235, 321
`fputs()`, 239-240, 321-322
`fread()`, 245-248, 322-323
`free()`, 138, 139, 434, 435-436
`freopen()`, 258-259, 323-324
`frexp()`, 399-400
`fscanf()`, 254-256, 324-325
`fseek()`, 232, 253-254, 325-326, 515

`fsetpos()`, 326-327
`ftell()`, 254, 327-328
`_func_` predefined identifier, 293-294
 Function(s), 7, 148-172, 719, 720
 arguments. *See* Arguments, function
 calling and returning sequences for, 694, 695
 declaration, implicit, 281, 296
 formal parameters of. *See* Parameters, formal
 general form of, 148
 implicit int rule and, 170-171
 inline, 172, 282-284
 passing entire structures to, 187-188, 189
 passing multidimensional arrays to, 103, 107-108
 passing pointers to, 150-152
 passing single-dimension arrays to 98-99
 passing structure members to, 186
 pointers to, 134-138
 prototype scope, 28
 prototypes, 166-169, 305-306, 706-707
 recursive, 164-166, 708
 returning from, 158-160
 returning pointers from, 162-163
 returning values from, 160-162
 scope, 28, 148-149
 stand-alone, 7, 33, 694
 termination, 442
 type declaration statement, old-style, 168-169
 used in expressions, 70, 160
 void, 163-164

`fwide()`, 475-477
`fwrite()`, 194, 245-248, 328-329

G

Generic pointer, 123
`getc()`, 236, 238, 329-330
`getch()`, 208-209
`_getch()`, 209, 529
`getchar()`, 206-208, 223, 330-331
`getche()`, 208-209
`_getche()`, 209
`getenv()`, 450-451
`GetMessage()` API function, 674-675
`gets()`, 154, 209-210, 331-332
`GetStockObject()` API function, 671
`gmtime()`, 422
 goto statement, 7, 28, 88, 148
 Graphic Device Interface (GDI), 662

H

Hashing, 575-579
 Headers, 169, 266, 305-308
 C89, table of, 306
 C99, table of, 307
 function-like macros in, 307-308
 Heap, 13, 138
 Hexadecimal constants, 38
 Hoare, C.A.R., 456, 508
`HUGE_VAL` macro, 384, 385
`HUGE_VALF` macro, 384
`HUGE_VALL` macro, 384
`hypot()`, 400

I

I macro, 484
 Icons, 660, 670
 IDE (integrated development environment), 689-690, 715

Identifiers, 20-21
 #if directive, 266-269, 270-271
 if statement
 conditional expression to control, 72
 general form of, 64-65
 and if-else-if ladder, 67-69
 nested, 66-67
 ? as alternative to, 69-71, 72
 #ifdef directive, 269-270
 #ifndef directive, 269-270
 ilogb(), 400-401
 _Imaginary data type, 18, 284, 484
 imaginary macro, 484
 _Imaginary_I macro, 484
 imaxdiv(), 490
 imaxdiv_t structure, 490
 Implicit int rule, 170-171, 281, 294-296
 #include directive, 265-266
 Increment operator (++), 44-45, 692-693, 700
 Incremental testing, 709
 Infinite loop, 82
 INFINITY macro, 384
 In-line code, 282-284
 for efficiency, 694-698
 inline keyword, 172, 265, 282-284, 698
 int data type, 18, 19, 20
 as default function return type, 170-171, 281, 294-295
 int16_t type, 297, 488
 int_fast32_t type, 297, 488
 int_least16_t type, 297, 488
 Integer
 format conversion functions, table of, 491
 types and macros defined by <stdint.h>, 488, 490
 Integral promotion, 59
 Integrated development environment (IDE), 689-690, 715
 Internal names, 21
 Interpreter(s)
 practical importance of, 714-715
 prescan, 745
 and structured languages, 718-719
 versus compilers, 9-10, 714, 715
 Interpreter, Little C, 715-787
 compiling and linking, 780
 expanding, 786-787
 expression parser for, 722-744
 improving, 785-786
 library functions, 776-780
 main() function, 748-749
 prescan, 745-748, 785
 production rules, 721
 specifications, 715-718
 intmax_t type, 297, 490
 intptr_t type, 490
 <inttypes.h> header, 293, 490
 I/O, console, 206-227
 basic functions for, table of, 210
 and characters, 206-209
 connection with file I/O, 257-258
 formatted, 212-227
 and strings, 209-212
 I/O, file, 230-259
 ANSI C versus UNIX, 230
 C versus C++, 230
 and characters, 235-236
 connection with console I/O, 257-258
 files in, 230, 231-232
 formatted with fprintf() and fscanf(), 254-256
 functions, table of common, 233

random-access, 253-254
 reading and writing blocks of data in, 245-248
 standard streams, 256-258
 streams in, 230, 231
 I/O functions, Standard C, 310-352
 wide-character, 474-477
 I/O, redirectable, 256, 257
 isalnum(), 354-355
 isalpha(), 355-356
 isblank(), 356-357
 iscntrl(), 357
 isdigit(), 358
 isfinite(), 384
 isgraph(), 358-359
 isgreater(), 384
 isgreaterequal(), 384
 isinf(), 384
 isless(), 384
 islessequal(), 385
 islessgreater(), 385
 islower(), 359-360
 isnan(), 384
 isnormal(), 384
 <iso646.h> header, 293
 isprint(), 360-361
 ispunct(), 361
 isspace(), 362
 isupper(), 362-363
 iswctype(), 473-474
 isxdigit(), 363-364
 Iteration statements, 76-87

J

Java, 715
 jmp_buf data type, 453
 Jump statements, 87-93

K

kbhit(), 89
 _kbhit(), 529
 Kernighan, Brian, 4
 Keywords
 added by C99, table of, 11
 C89, table of, 10
 extended, common, 10

L

Label, 28, 88
 labs(), 451
 Languages, characteristics of computer, 5-9
 LC_ALL macro, 427, 428
 LC_COLLATE macro, 427, 428
 LC_CTYPE macro, 428
 LC_MONETARY macro, 428
 LC_NUMERIC macro, 428
 LC_TIME macro, 428
 lconv structure, 418, 423-424
 ldexp(), 401
 ldiv(), 440, 452
 ldiv_t type, 440, 452
 lgamma(), 402
 Library
 definition of, 15
 dynamic link (DLL), 304
 files versus object files, 305
 functions, redefining, 308
 standard, 11-12, 302, 304, 305
 LIFO (last-in, first-out), 531
 #line directive, 271-272
 __LINE__ predefined macro, 271-272, 273
 Line-buffered input, 207
 Linked lists
 doubly, 541-553

- singly, 536-541
- for sparse arrays, 565-569
- Linker
 - definition of, 15
 - and library files, 305
 - operation of, 302-304, 308
 - overlay, 281-282
- Linking, 11, 302-304
 - dynamic, 304
 - and global variables, 32-33
 - library and, 13, 308
- Lists, linked. *See* Linked lists
- Literals, 37
 - compound, 290-291
- llabs(), 451-452
- lldiv(), 453
- lldiv_t structure, 453
- llrint(), 402
- llround(), 402-403
- LoadCursor() API function, 670-671
- LoadIcon() API function, 670
- LoadImage() API function, 670
- localeconv(), 418, 423-425
- <locale.h> header, 418
- localtime(), 425-426
- log(), 403
- log1p(), 404
- log2(), 405
- log10(), 404-405
- logb(), 405
- Logical operators, 46-48
 - truth table for, 46
- long long modifier, 18, 285
- long modifier, 18, 19
- longjmp(), 453-454, 604
- Loops, 7, 64
 - do-while, 86-87
 - for. *See* for loop
 - infinite, 82
 - message (Windows), 665, 674-675
 - while, 83-86
 - with no bodies, 82-83
- LRESULT data type, 664
- lrint(), 406
- lround(), 406
- lvalue, 40

M

- Machine code, 9
- Macro
 - function-like, 264-265, 287, 307-308
 - name, 262
 - predefined, 273-274
 - replacement, 262
 - type-generic math, 490, 492-493
- main(), 11, 720
 - argc and argv as arguments to, 155-158
 - return value from, 164
 - using void in parameter list of, 158
 - and Windows programs, 664
- MAKE, 685-689
- Make file, general form of, 686
- malloc(), 138-139, 140, 141, 434, 436-437, 701-702
- <math.h> header, 384
- math_errhandling macro, 384, 385
- MATH_ERRNO macro, 384, 385
- MATH_ERREXCEPT macro, 384, 385
- MB_CUR_MAX macro, 440
- mblen(), 454-455
- mbstate_t type, 472, 480
- mbstowcs(), 455-456
- mbtowc(), 456
- memchr(), 364-365

- memcmp(), 365-366
- memcpy(), 282, 366-367
- memmove(), 367
- Memory
 - dynamic allocation of, 138-142, 434-438
 - regions of, 13-14
- memset(), 368
- Message(s), 664-665
 - loop, 665, 674-675
- mktime(), 426-427
- modf(), 406-407
- Modula-2, 5, 6, 8
- Mouse, 661
 - cursor, loading, 670-671
- MSG structure, 666, 674, 675
- Multitasking, preemptive, 662

N

- NAN macro, 384
- nan(), 407
- NDEBUG macro, 442
- nearbyint(), 407-408
- nextafter(), 408
- nexttoward(), 408
- NOT logical operator (!), 47
- NULL macro, 132, 232, 440
- Numeric constants, 37-38

O

- Object code, 9, 15, 302
 - synchronizing, with source code, 685-689
- Octal constants, 38
- Offset, 303
- One's complement operator, 49, 52-53
- Operator(s), 40-58

- arithmetic, 43-46
- assignment, 40
- bitwise, 48-53
- comma, 56
- compile-time, 55-56, 270-271
- dot and arrow, 56-57
- pointer, 53-55, 121
- precedence summary table, 58
- preprocessor, 272-273
- relational and logical, 46-48
- ternary, 53, 69-71

OR

- bitwise operator (|), 49, 50
- logical operator (||), 47
- Overlays, 303-304

P

- Parameters, formal, 25-26, 148, 149
 - declarations, classic versus modern, 171-172
 - variable number of, 169-170
- Parameters, reference, 152
- Parity bit, 50
- Parser, recursive descent, 582, 586-604
 - definition of, 587, 723
 - Little C, 722-744
 - syntax checking in, 604
- Parser, table-driven, 584, 722-723
- Pascal, 5, 6, 8
- perror(), 332
- POINT structure, 674
- Pointer(s), 120-146
 - accessing arrays with, 128-129
 - arithmetic, 109, 124-125, 694
 - arrays of, 129-130, 572-575
 - assignments, 122-123
 - base type of, 54, 121, 124, 125
 - comparisons, 126-127, 144-145

conversions, 123-124
 definition of, 53, 120
 file, 232, 310
 generic, 123
 indexing, 108-110
 initializing, 131-134, 143-144, 145-146, 702
 multiple indirection with, 130-131
 operators, 53-55, 121
 passed to functions, 150-152
 problems with, 143-146
 restrict-qualified, 142-143, 282
 returned from functions, 162-163
 structure, 188-192
 to arrays, generating, 97-98
 to functions, 134-138
 void*, 123, 124, 138, 139, 162
 Portability, 5
 and operating system dependencies, 699
 using #define for, 698-699
 using sizeof to ensure, 201-203, 699
 PostQuitMessage() API function, 675
 pow(), 409
 #pragma directive, 272
 _Pragma operator, 272, 288
 Pragmas, built-in, 288
 Preprocessor directives, 262-272
 Preprocessor operators, 272-273
 printf(), 212-221, 292-293, 332-336
 and disk file I/O, 257
 format specifiers, table of, 213, 334
 return value of, 161, 212, 333
 Problem solving, AI-based, 606-656

backtracking and, 617
 and combinatoric explosions, 608-610
 and finding multiple solutions, 637-645
 and finding optimal solution, 645-652
 terminology, 606-608
See also Search (AI)
 Program(s), 719-720
 creating executable form of, 13
 general form of, 10-11, 12
 systems, 9
 See also Software engineering
 Programming for efficiency, 692-698
 Prototypes, function, 166-169, 305-306, 706-707
 Pseudocode, 681
 Push and pop operations, 531, 694
 putc(), 235, 336-337
 putchar(), 206-207, 256, 337
 puts(), 209, 210, 337-338

Q

qsort(), 456-458, 498
 Queues, 522-530
 Quicksort, 456, 508-511

R

raise(), 458
 rand(), 459
 RAND_MAX macro, 440, 459
 Random-access disk files, sorting, 515-518
 Random numbers, generating, 459, 460-461
 realloc(), 434, 437-438
 Recursion, 164-166, 708

register storage class specifier, 35-36
 for efficiency, using, 693-694
 RegisterClassEx() API function, 671
 Relational operator(s), 46-48
 Relocatable format, 11-12
 remainder(), 409-410
 remove(), 243-244, 338-339
 remquo(), 410
 rename(), 339
 restrict type qualifier, 28, 142-143, 282, 286, 310, 354
 return statement, 87, 296
 using to return from a function, 158-160
 using to return a value, 160-162
 Returning sequence, 694, 695
 rewind(), 240-241, 340
 Richards, Martin, 4
 rint(), 410-411
 Ritchie, Dennis, 4
 round(), 411
 Run time, 15
 allocating memory at, 138-142
 *rvalue, 40

S

Scalar, 65
 scalbln(), 411
 scalbn(), 412
 scanf(), 221-227, 292-293, 340-345
 format specifiers, table of, 222, 341
 Scanset, 224-225, 343
 Scopes, C, 27-28
 Search (AI)
 adding heuristics to, 626-628
 breadth-first, 625-626
 depth-first, 613-625
 evaluating a, 610-611

 exhaustive, 610, 637, 646
 hill-climbing, 628-635
 least-cost, 635-636
 and node removal, 639-645
 and path removal, 638-639
 technique, choosing, 636-637
 Searching, 518-520
 as AI problem-solving technique, 606
 SEEK_CUR macro, 232, 253, 325
 SEEK_END macro, 232, 253, 325
 SEEK_SET macro, 232, 253, 254, 325
 Selection statements, 64-76
 setjmp(), 459-460, 604
 <setjmp.h> header, 453, 459
 setlocale(), 427-428
 setvbuf(), 345-346
 Shaker sort, 503
 Shell sort, 506-508
 short modifier, 18, 19
 ShowWindow() API function, 673-674
 SIG_DFL macro, 460
 SIG_ERR macro, 460
 SIG_IGN macro, 460
 SIGABRT macro, 458
 SIGFPE macro, 458
 SIGILL macro, 458
 SIGINT macro, 458
 Sign flag, 20
 signal(), 460
 <signal.h> header, 458
 signbit(), 384
 signed modifier, 18, 19
 SIGSEGV macro, 458
 SIGTERM macro, 458
 sin(), 161, 412-413
 sinh(), 413-414
 sizeof operator, 55-56, 201-203, 285, 699

size_t data type, 55-56, 232, 310, 354, 440, 472
 snprintf(), 346
 Software engineering
 and creating bulletproof functions, 682-685
 and top-down design method, 680-682
 using an IDE for, 689-690
 using MAKE in, 685-689
 Sorting, 496-518
 algorithms, judging, 499-500
 arrays, 498-511
 with bubble sort, 500-503
 by exchange, 499, 500, 508
 by insertion, 499, 505-506
 key, 498
 with quicksort, 508-511
 random-access disk files, 515-518
 by selection, 499, 504-505
 with shaker sort, 503
 with Shell sort, 506-508
 strings, 512-513
 structures, 513-515
 Source code, 302
 definition of, 15
 synchronizing, with object code, 685-689
 Sparse arrays, 564-580
 binary tree, 569-571
 choosing an approach for, 580
 hashing and, 575-579
 linked-list, 565-569
 pointer array, 572-575
 sprintf(), 347
 sqrt(), 161, 414
 srand(), 460-461
 sscanf(), 347-348

Stack
 as a data engine, 531-535
 and local variables, 24
 as memory region, 13
 overruns, 708
 Statements, 719, 720
 block, 93-94
 expression, 93
 iteration, 76-87
 jump, 87-93
 label, 64, 73
 null, 93
 selection, 64-76
 static storage class specifier, 24, 33-35, 149, 286
 <stdarg.h> header, 468
 <stdbool.h> header, 284, 293, 493
 __STDC__ predefined macro, 273, 274
 __STDC_HOSTED__ predefined macro, 274, 289
 __STDC_IEC_559__ predefined macro, 289
 __STDC_IEC_559_COMPLEX__ predefined macro, 279
 __STDC_ISO_10646__ predefined macro, 289
 __STDC_VERSION__ predefined macro, 289
 <stddef.h> header, 37
 stderr standard stream, 256, 257, 442
 stdin standard stream, 256-259
 <stdint.h> header, 293, 297, 488, 490
 <stdio.h> header, 206, 232, 305, 310
 <stdlib.h> header, 90, 138, 434, 472
 stdout standard stream, 256-259
 Storage class specifiers, 30-36
 strcat(), 100, 368-369
 strchr(), 100, 369-370

strcmp(), 80, 100, 101, 135-136, 370-371, 512, 513
 strcoll(), 371
 strcpy(), 100, 372, 513
 strcspn(), 372-373
 Streams, 230-231
 flushing, 231, 244
 Streams, standard, 256-259
 strerror(), 373
 strftime(), 428-431
 format commands, table of, 429-430
 String(s)
 as array, 96, 99-101
 arrays of, 106-107
 in console I/O, 209-212, 223-224
 constants, 38, 100
 in file I/O, 239-240
 sorting, 512-513
 table, 134, 512
 string class (C++), 100
 String handling functions, 100-101, 365-381
 wide-character, 477-478
 <string.h> header, 100, 354
 strlen(), 29-30, 85, 100, 373-374
 strncat(), 374-375
 strncmp(), 375-376
 strncpy(), 376-377
 strpbrk(), 377
 strrchr(), 377-378
 strspn(), 378-379
 strstr(), 100, 379-380
 strtod(), 461-463
 strof(), 463
 strtok(), 380-381
 strtol(), 463-464
 strol(), 464
 stroll(), 465

strtoul(), 465-466
 strtoull(), 466
 strxfrm(), 381
 struct keyword, 174, 201
 Structure(s)
 accessing member of, 176-177
 arrays and structures
 within, 192
 arrays of, 178-185
 assignments, 177
 declaration, 174, 176
 designated initializers and, 291-292
 and flexible array members, 291
 to functions, passing entire, 187-188, 189
 members to functions,
 passing, 186
 pointers, 188-192
 and sizeof, 201-202
 sorting, 513-515
 variable, declaring, 174-176
 switch statement, 72-76, 89, 90
 system(), 467
 Systems program, 9

T

tan(), 414-415
 tanh(), 415-416
 Ternary operator (?), 53, 69-71, 72
 tgamm(), 416
 <tgmath.h> header, 293, 492
 Thompson, Ken, 4
 Time
 broken-down, 418
 calendar, 418
 time(), 431-432
 __TIME__ predefined macro, 273, 274

<time.h> header, 418
 time_t type, 418
 tm structure, 418
 tmpfile(), 348-349
 tmpnam(), 349-350
 Token, 584, 723
 tolower(), 381-382
 toupper(), 382
 towctrans(), 474
 TranslateMessage() API
 function, 675
 Translation unit, 27
 True and false in C, 46, 64
 true macro, 284, 493
 trunc(), 416
 Two's complement, 20
 Type conversions in assignments,
 40-42
 Type promotion, 59-60, 297
 Type qualifiers, 28-30
 typedef statement, 30, 203

U

uintmax_t type, 297, 490
 uintptr_t type, 490
 #undef directive, 270
 ungetc(), 350
 union keyword, 193, 201
 Unions, 174, 193-195
 and sizeof, 201-203
 UNIX, 4
 unsigned modifier, 18, 19
 UpdateWindow() API function, 674

V

va_arg(), 467-469
 __VA_ARGS__, 287-288
 va_copy(), 467-469

va_end(), 467-469
 va_list data type, 468
 Variable, 21-27
 automatic, 22
 declaration versus definition of,
 31, 32
 declaration within for loop, 83
 initializing, 36-37
 pointer, 53, 54-55, 121
 storage class specifiers for,
 30-36
 type qualifiers, 28-30
 Variables, global, 26-27, 148
 extern used with, 32-33
 static, 34-35
 Variables, local, 21-26
 static used with, 24, 33-34,
 35, 149
 va_start(), 467-469
 vfprintf(), 351-352
 vfscanf(), 352
 void data type, 18, 19
 volatile type qualifier, 30, 286
 vprintf(), 351-352
 vscanf(), 352
 vsnprintf(), 351-352
 vsprintf(), 351-352
 vsscanf(), 352

W

<wchar.h> header, 293
 wchar_t type, 37, 310, 354, 440, 472
 WCHAR_MAX, 472
 WCHAR_MIN, 472
 wcstombs(), 469-470
 wctomb(), 470
 wctrans(), 474
 wctrans_t type, 472
 wctype(), 473-474

<wctype.h> header, 293, 472
 wctype_t type, 472
 WEOF, 472
 while loop, 83-86, 92
 Wide-character functions, 472-482
 Win32 API, 662
 WINAPI calling convention, 664
 Window
 class, 665, 666
 components of, 662-663
 creating, 672-673
 displaying, 673-674
 function, 664-665, 666, 675
 Windows
 application basics, 664-666
 and C, 660
 data types, 665-666
 desktop model, 661
 and DLLs, 304
 interaction with programs, 662,
 663-664
 and the mouse, 661
 and multitasking, 662, 669

naming conventions, 676-677
 programming overview,
 660-662
 skeleton, 666-675
 Windows 2000 Programming from
 the Ground Up (Schildt), 660
 WINDOWS.H header file, 665, 668
 WinMain(), 664, 665, 666, 669,
 674, 675
 wint_t type, 472
 WM_DESTROY message, 675
 WM_QUIT message, 675
 WNDCLASSEX structure, 666, 668,
 669-670

X

XOR
 bitwise operator (^), 49, 50
 logical operation, 46-48