

# C: The Complete Reference

73.82 9.10  
5.1

# C: The Complete Reference

Second Edition

Herbert Schildt

Osborne McGraw-Hill

Berkeley New York St. Louis San Francisco  
Auckland Bogotá Hamburg London Madrid  
Mexico City Milan Montreal New Delhi Panama City  
Paris São Paulo Singapore Sydney  
Tokyo Toronto

840017

Osborne McGraw-Hill  
2600 Tenth Street  
Berkeley, California 94710  
U.S.A.

DS71 / 24

For information on translations and book distributors outside of the U.S.A., write to Osborne McGraw-Hill at the above address.

A complete list of trademarks appears on page 803.

### **C: The Complete Reference, Second Edition**

Copyright © 1987, 1990 by McGraw-Hill, Inc. All rights reserved. Printed in the United States of America. Except as permitted under the Copyright Act of 1976, 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, with the exception that the program listings may be entered, stored, and executed in a computer system, but they may not be reproduced for publication.

34567890 DOC 90

ISBN 0-07-881538-X

Information has been obtained by Osborne McGraw-Hill from sources believed to be reliable. However, because of the possibility of human or mechanical error by our sources, Osborne McGraw-Hill, or others, Osborne McGraw-Hill does not guarantee the accuracy, adequacy, or completeness of any information and is not responsible for any errors or omissions or the results obtained from use of such information.

1100110

Most of the time, writing a book is challenging, tedious, exciting, frustrating, and a lot of work all rolled into one. But the creation of *C: The Complete Reference, Second Edition*, had an extra, almost intangible component. I had been given the mission of creating a complete reference guide to the C programming language. Not just *any* book, but the *complete* book. Whenever I write a book, I try to be thorough, but in this book I had to be even more thorough. It was the word "complete" that kept haunting me. I knew that programmers from around the world would be buying my book because it was the *complete* reference. Finally, I realized, as I'm sure you must, that no one book on C can hope to cover all of its nuances, tricks, and uses. The C language is just too rich and powerful. But I do say this: The writing of *C: The Complete Reference, Second Edition*, was a labor of love. I am confident that you will be pleased with the result.

### Is This Book for You?

---

This C reference is designed for all C programmers, regardless of their experience level. It does assume, however, a reader able to create at least a simple C program. If you are just learning C, this book will make an excellent companion to any C tutorial and serve as a source of answers to your specific questions.

### What's New In the Second Edition

---

For the most part, I have left the basic structure of the book unchanged from the first edition. A couple of chapters have been moved, but the presentation of material within each chapter reflects the original book. Also, much new material has been added, including expanded coverage of graphics.

The major differences between the first and the second editions are caused by conformance to the new ANSI C standard. When the first edition was written, the ANSI C standard, still in its infancy, was being continually changed. Also, at that time no compilers conformed closely to the evolving standard. Therefore, most of the original edition reflected the more common de facto C standard as defined by Kernighan and Ritchie. However, now that the standard is stable, and given the fact that all major C compilers comply with the standard, it seemed time to switch to the ANSI C standard. (Important points and differences between the old de facto standard and the new ANSI standard are still covered by this book, however, so you don't have to be using an ANSI C standard compiler to utilize the book.)

The major changes caused by conformance to the ANSI C standard are function prototyping, which is used throughout the book, and inclusion of all header files. In older versions of C, most header files were not needed for programs to run correctly. However, in order to take advantage of function prototyping, all header files must be included because they contain the prototypes of the standard library functions.

Another change caused by the adoption of the ANSI C standard is that the modern form of function declaration is used. (The older—sometimes called “classic”—form is still explained in detail, but the code examples use the modern form.)

## What's Inside

---

This book covers in detail all aspects of the C language and its libraries. It includes both the old de facto C standard and the ANSI C standard. However, the main emphasis is on the ANSI C standard. The book is divided into five parts, covering

- The C language
- The C libraries
- Common algorithms and applications
- The C programming environment
- C++: C's newest direction

7100240

Part One provides a thorough discussion of the keywords, preprocessor directives, and features of the C language.

Part Two discusses the standard C library. Both the K&R and ANSI standard library functions are covered. Also included are some functions common to the DOS programming environment, including DOS calls and graphics. For DOS-specific functions, Microsoft C was chosen for the examples.

Part Three covers some of the more common and important algorithms and applications that all C programmers should have in their toolbox. You will find Chapter 22, which deals with AI problem-solving techniques, particularly interesting.<sup>1</sup>

Part Four covers the C programming environment including such things as interfacing to assembly code, efficiency, porting, and debugging.

Part Five examines C's newest direction, C++. C++ is a superset of C designed to handle the rigorous needs of large projects and to aid in program verification.

There are many, many working functions and programs contained in this book. If you are like me, you like having a lot of routines to draw upon but hate entering them into the computer. It always seems that I type something wrong and spend hours trying to get the program to work. For this reason, I am offering on disk the source code to all the functions and programs contained in this book for \$24.95. Just fill in the order blank on the next page and mail it, along with your payment, to

Herbert Schildt  
RR 1, Box 130  
Mahomet IL 61853

Or if you're in a hurry, just call (217) 586-4021 (the number of my consulting office) and place your order by telephone (VISA and MasterCard accepted).

—HS

<sup>1</sup>For a wider assortment of C applications, refer to my book *Advanced C, Second Edition* (Berkeley, Calif.: Osborne/McGraw-Hill, 1988). For those interested in AI, see my book *Artificial Intelligence Using C* (Berkeley, Calif.: Osborne/McGraw-Hill, 1987).

## Order Form

---

Please send me \_\_\_\_\_ copies, at \$24.95 each, of the programs in *C: The Complete Reference, 2nd Edition*. Foreign orders, please add \$5 shipping and handling.

\_\_\_\_\_  
Name

\_\_\_\_\_  
Address

\_\_\_\_\_  
City

\_\_\_\_\_  
State

\_\_\_\_\_  
ZIP

\_\_\_\_\_  
Telephone

Disk size: 5 1/4 \_\_\_\_\_ 3 1/2 \_\_\_\_\_

Method of payment: check \_\_\_\_\_ VISA \_\_\_\_\_ MC \_\_\_\_\_

Credit card number \_\_\_\_\_

Expiration date \_\_\_\_\_

Signature \_\_\_\_\_

Send to:

Herbert Schildt  
RR 1, Box 130  
Mahomet, IL 61853

or phone: (217) 586-4021

This is solely the offering of the author. Osborne/McGraw-Hill takes no responsibility for the fulfillment of this offer. Please allow four to six weeks for delivery.

	Preface	xiii
<b>PART ONE</b>	<b>The C Language</b>	<b>1</b>
<b>ONE</b>	<b>An Overview of C</b>	<b>3</b>
	The Origins of C	3
	C Is a Middle-Level Language	4
	C Is a Structured Language	5
	C Is a Programmer's Language	7
	Compilers Versus Interpreters	9
	The Form of a C Program	10
	The Library and Linking	11
	Separate Compilation	13
	Compiling a C Program	13
	C's Memory Map	14
	A Review of Terms	15
<b>TWO</b>	<b>C Expressions</b>	<b>17</b>
	The Five Basic Data Types	17
	Modifying the Basic Types	19
	Identifier Names	20
	Variables	21
	Access Type Modifiers	28
	Storage Class Type Specifiers	31
	Variable Initializations	38
	Constants	38
	Operators	41
	Expressions	60
<b>THREE</b>	<b>Program Control Statements</b>	<b>65</b>
	True and False in C	66
	Selection Statements	66
	Iteration Statements	78
	Jump Statements	89
	Expression Statements	95
	Block Statements	95
<b>FOUR</b>	<b>Arrays and Strings</b>	<b>97</b>
	Single-Dimension Arrays	97
	Generating a Pointer to an Array	99
	Passing Single-Dimension Arrays to Functions	100



Strings	101
Two-Dimensional Arrays	103
Multidimensional Arrays	108
Indexing Pointers	109
Array Initialization	112
A Tic-Tac-Toe Example	114

## **FIVE**

### **Pointers 119**

What Are Pointers?	119
Pointer Variables	120
The Pointer Operators	121
Pointer Expressions	122
Pointers and Arrays	126
Multiple Indirection	129
Initializing Pointers	130
Pointers to Functions	132
C's Dynamic Allocation Functions	135
Problems with Pointers	140

## **SIX**

### **Functions 145**

The General Form of a Function	145
Scope Rules of Functions	146
Function Arguments	147
argc and argv—Arguments to main()	154
The return Statement	157
Functions That Return Noninteger Values	160
Function Prototypes	163
Returning Pointers	164
Functions of Type void	165
What Does main() Return?	166
Recursion	167
Declaring Variable Length and Type Parameter Lists	169
Classic Versus Modern Function Parameter Declarations	169
Implementation Issues	171
Libraries and Files	172

<b>SEVEN</b>	<b>Structures, Unions, Enumerations, and User-Defined Types</b>	<b>175</b>
	Structures 175	
	Arrays of Structures 179	
	Passing Structures to Functions 186	
	Structure Pointers 189	
	Arrays and Structures within Structures 192	
	Bit-Fields 193	
	Unions 196	
	Enumerations 198	
	Using sizeof to Ensure Portability 201	
	typedef 203	
<b>EIGHT</b>	<b>Console I/O</b>	<b>205</b>
	Reading and Writing Characters 206	
	Reading and Writing Strings 208	
	Formatted Console I/O 211	
	printf() 211	
	scanf() 219	
<b>NINE</b>	<b>File I/O</b>	<b>227</b>
	ANSI C I/O Versus UNIX I/O 227	
	Streams and Files 228	
	Streams 228	
	Files 229	
	File System Basics 230	
	fread() and fwrite() 243	
	fseek() and Random-Access I/O 249	
	fprintf() and fscanf() 250	
	The Standard Streams 251	
	The UNIX-like File System 254	
<b>TEN</b>	<b>The C Preprocessor and Comments</b>	<b>261</b>
	The C Preprocessor 261	
	#define 262	
	#error 265	
	#include 265	
	Conditional Compilation Directives 266	
	#undef 270	
	#line 270	

	#pragma 271	
	The # and ## Preprocessor Operators 271	
	Predefined Macro Names 272	
	Comments 273	
<b>PART TWO</b>	<b>The C Standard Library</b>	<b>275</b>
<b>ELEVEN</b>	<b>Linking, Libraries, and Header Files</b>	<b>277</b>
	The Linker 277	
	The C Standard Library 282	
	Header Files 283	
	Redefinition of Library Functions 285	
<b>TWELVE</b>	<b>I/O Functions</b>	<b>287</b>
<b>THIRTEEN</b>	<b>String and Character Functions</b>	<b>345</b>
<b>FOURTEEN</b>	<b>Mathematical Functions</b>	<b>373</b>
<b>FIFTEEN</b>	<b>Time, Date, and Other System-Related Functions</b>	<b>389</b>
<b>SIXTEEN</b>	<b>Dynamic Allocation</b>	<b>439</b>
<b>SEVENTEEN</b>	<b>Screen and Graphics Functions</b>	<b>459</b>
<b>EIGHTEEN</b>	<b>Miscellaneous Functions</b>	<b>491</b>
<b>PART THREE</b>	<b>Algorithms and Applications</b>	<b>519</b>
<b>NINETEEN</b>	<b>Sorting and Searching</b>	<b>521</b>
	Sorting 521	
	Choosing a Sort 536	
	Sorting Other Data Structures 537	
	Sorting Random-Access Disk Files 539	
	Searching 542	
<b>TWENTY</b>	<b>Queues, Stacks, Linked Lists, and Trees</b>	<b>545</b>
	Queues 546	
	The Circular Queue 551	

Stacks 554  
Linked Lists 559  
Singly Linked Lists 559  
Doubly Linked Lists 564  
A Mailing-List Example 569  
Binary Trees 574

**TWENTY-ONE Sparse Arrays 583**

The Linked-List Sparse Array 584  
The Binary-Tree Approach to Sparse  
Arrays 587  
The Pointer-Array Approach to Sparse  
Arrays 590  
Hashing 593  
Choosing an Approach 599

**TWENTY-TWO Expression Parsing and Evaluation 601**

Expressions 602  
Dissecting an Expression 603  
Expression Parsing 606  
A Simple Expression Parser 608  
Adding Variables to the Parser 613  
Syntax Checking in a Recursive-Descent  
Parser 621

**TWENTY-THREE AI-Based Problem Solving 623**

Representation and Terminology 624  
Combinatorial Explosions 625  
Search Techniques 628  
Evaluating a Search 628  
A Graphic Representation 630  
The Depth-First Search 631  
The Breadth-First Search 641  
Adding Heuristics 643  
The Hill-Climbing Search 644  
The Least-Cost Search 650  
Choosing a Search Technique 652  
Finding Multiple Solutions 652  
Finding the "Optimal" Solution 659  
Back to the Lost Keys 664

<b>TWENTY-FOUR</b>	<b>Using System Resources</b>	<b>669</b>
	The 8086 Family of Processors	669
	The 8086 Interrupts and DOS	670
	Accessing System Resources in the BIOS	672
	Using DOS to Access System Functions	684
	Final Thoughts on Using System Resources	689
<b>TWENTY-FIVE</b>	<b>Graphics</b>	<b>691</b>
	Modes and Palettes	692
	Writing Pixels	693
	Drawing Lines	697
	Drawing and Filling Boxes	701
	Putting It All Together	703
<b>PART FOUR</b>	<b>Software Development Using C</b>	<b>715</b>
<b>TWENTY-SIX</b>	<b>Interfacing to Assembly-Language Routines</b>	<b>717</b>
	Assembly-Language Interfacing	717
	The Calling Conventions of a C Compiler	718
	The Calling Conventions of Microsoft C	719
	Creating an Assembly-Code Function	720
	Creating An Assembly-Code Skeleton	729
	Using asm	732
	When to Code in Assembler	733
<b>TWENTY-SEVEN</b>	<b>Software Engineering Using C</b>	<b>735</b>
	Top-Down Design	735
	Bulletproof Functions	738
	Function Prototypes	741
	LINT and MAKE	741
<b>TWENTY-EIGHT</b>	<b>Efficiency, Porting, and Debugging</b>	<b>749</b>
	Efficiency	749
	Porting Programs	757
	Debugging	760
	The Art of Program Maintenance	771

**PART FIVE**

**New Directions**

**TWENTY-NINE**

**C + +**

**777**

Data Abstraction 778

Objects 780

Comments 780

Classes 781

Function Overloading 789

Operator Overloading 791

Other Features of C + + 795

**APPENDIX**

**Differences Between K&R C and ANSI  
Standard C**

**797**

Keyword Deletions 797

Keyword Extensions 798

Passing Structures 800

Function Prototypes 800

Modern Versus Classic Function Declarations  
801

Standard Libraries 801

Additional Preprocessor Directives 802

**Index**

**805**

# The C Language

An Overview of C  
C Expressions  
Program Control Statements  
Arrays and Strings  
Pointers  
Functions  
Structures, Unions, Enumerations,  
and User-Defined Types  
Console I/O  
File I/O  
The C Preprocessor and Comments

# P A R T O N E

The first part of this reference guide presents a thorough discussion of the C programming language. Chapter 1 provides a quick overview of the C language—the more knowledgeable programmer may wish to skip directly to Chapter 2. Chapter 2 examines C's built-in data types, variables, operators, and expressions. Next, Chapter 3 presents program control statements. Chapter 4 discusses arrays and strings. Chapter 5 looks at pointers. Chapter 6 deals with functions, and Chapter 7 discusses structures, unions, and user-defined types. Chapter 8 examines console I/O. Chapter 9 covers file I/O, and Chapter 10 discusses the C preprocessor and comments.

The material in this section (and most of the material in the book) reflects the ANSI standard for C. However, the original C standard as derived from the UNIX version 5 C is also covered, and important differences are noted. The book covers both ANSI and the original C to ensure that you find information relevant to your C programming environment.





## An Overview of C

- The Origins of C
- C Is a Middle-Level Language
- C Is a Structured Language
- C Is a Programmer's Language
- Compilers Versus Interpreters
- The Form of a C Program
- The Library and Linking
- Separate Compilation
- Compiling a C Program
- C's Memory Map

The purpose of this chapter is to present an overview of the C programming language, its origins, its uses, and its underlying philosophy. This chapter is mainly for newcomers to C.

### The Origins of C

---

C was invented and first implemented by Dennis Ritchie on a DEC PDP-11 that used the UNIX operating system. C is the result of a development process that started with an older language called BCPL, which is still in use primarily in Europe. BCPL was developed by Martin Richards, and it influenced a language called B, which was invented by Ken Thompson. B led to the development of C in the 1970s.

For many years, the de facto standard for C was the version supplied with the UNIX version 5 operating system. It is described in *The C Programming Language* by Brian Kernighan and Dennis Ritchie (Englewood Cliffs, N.J.: Prentice-Hall, 1978). With the popularity of microcomputers, a large number of C implementations were created. In a near miracle, the source codes of most of these implementations were