

Over 150,000

TEACH YOURSELF

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

-ACM Computing Reviews

C



Third
Edition

The Most Successful and Proven Method of Learning C

- Master all C fundamentals •
- Build your skills with hundreds of examples and exercises •
- Take charge of advanced features and techniques •

Herbert Schildt

Best-Selling C/C++ Author with More Than 1.5 Million Books Sold

Teach



yourself

C

Third Edition

Herbert Schildt

DITION-2003-2004

Osborne McGraw-Hill

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

Publisher
Brandon A. Nordin

Editor-in-Chief
Scott Rogers

Acquisitions Editor
Wendy Rinaldi

Project Editor
Heidi Poulin
Ann Sellers

Technical Editor
Gregg Guntle

Copy Editor
Erik Van Eaton

Proofreader
Karen Mead

Indexer
Sheryl Schildt
Computer Designers

Roberta Steele
Sylvia Brown

Illustrator
Roberta Steele

Series Design
Peter Hancik

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

For information on translations or book distributors outside U.S.A. or to arrange bulk purchase discounts for sales Promotions, premiums, or fundraisers, please contact Osborne McGraw-Hill at the above address.

Copyright © 1997 by The McGraw-Hill Companies. All Rights reserved, printed in the United States of America. Except as permitted under the Copyright Act of 1996, no part of this publication may be reproduced or distributed in any form or

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, executed in a computer system, but they may not be reproduced for publication.

4567890 1000 99

ISBN 0-07-882311-0

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

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 nearly two million copies worldwide and have been translated into all major foreign languages. He is the author of numerous best-sellers, including *c: The Complete Reference*, *C++: The Complete Reference*, *C++ from the Ground Up*, *FMCI programming from the Ground Up*, *Windows 95 Programming in C and C++*, *Windows NT 4 Programming from the Ground Up*, and many others. He is also a member of both the ANSI C and C++ standardization committees.

Contents

Preface, xi

for Further Study, xvii

1. C Fundamentals- - - 1

- 1.1 UNDERSTAND THE COMPONENTS OF AC PROGRAM, 2
- 1.2 CREATE AND COMPILE A PROGRAM, 7
- 1.3 DECLARE VARIABLES AND ASSIGN VALUES, 10
- 1.4 INPUT NUMBERS FROM THE KEYBOARD, 15
- 1.5 PERFORM CALCULATIONS USING ARITHMETIC EXPRESSIONS, 17
- 1.6 ADD COMMENTS TO A PROGRAM, 20
- 1.7 WRITE YOUR OWN FUNCTIONS, 23
- 1.8 USE FUNCTION ARGUMENTS, 32
- 1.9 REMEMBER THE C KEYWORDS, 32
- 1.10 REMEMBER THE C KEY WORDS, 35

2. Introducing C's Program Control Statements - - - 39

- 2.1 BECOM FAMILLIAR WITH THE if, 41
- 2.2 ADD THE else, 44
- 2.3 CREATE BLOCKS OF CODE, 46
- 2.4 USE THE for LOOP, 49
- 2.5 SUBSTITUTE C'S INCREMENT AND DECREMENT OPERATORS
- 2.6 EXPAND printf()'S CAPABILITIES, 58
- 2.7 PROGRAM WITH C'S RELATIONAL AND LOGICAL OPERATORS, 61

3. More C Program Control Statements - - - 69

- 3.1 INPUT CHARACTERS, 70
- 3.2 NEST if STATEMENTS, 75
- 3.3 EXAMINE for LOOP VARIATIONS, 79
- 3.4 UNDERSTAND C'S while LOOP, 82
- 3.5 USE THE do LOOKKP, 84
- 3.6 CREATE NESTED LOOPS, 87

- 3.7 USE break TO EXIT A LOOP. 89
- 3.8 KNOW WHEN TO USE THE continue STATEMENT. 92
- 3.9 SELECT AMONG ALTERNATIVES WITH THE switch STATEMENT. 94
- 3.10 UNDERSTAND THE goto STATEMENT. 100

- 4. A Closer look at Data Types, Variables, and Expressions --- 105**
- 4.1 USE C'S DATA-TYPE MODIFIERS. 107
- 4.2 LEARN WHERE VARIABLES ARE DECLARED. 107
- 4.3 TAKE A CLOSER LOOK AT CONSTANTS. 109
- 4.4 INITIALIZE VARIABLES. 123
- 4.5 UNDERSTAND TYPE CONVERSIONS IN EXPRESSIONS. 126
- 4.6 UNDERSTAND TYPE CONVERSIONS IN ASSIGNMENTS. 129
- 4.7 PROGRAM WITH TYPE CASTS. 132

- 5. Exploring Arrays and Strings -- 137**
- 5.1 DECLARE ONE-DIMENSIONAL ARRAYS. 139
- 5.2 USE STRINGS. 145
- 5.3 CREATE MULTIDIMENSIONAL ARRAYS. 151
- 5.4 INITIALIZE ARRAYS. 154
- 5.5 BUILD ARRAYS OF STRINGS. 159

- 6. Using Pointers --- 165**
- 6.1 UNDERSTAND POINTER BASICS. 167
- 6.2 LEARN RESTRICTIONS TO POINTER EXPRESSIONS. 172
- 6.3 USE POINTERS WITH ARRAYS. 176
- 6.4 USE POINTERS TO STRING CONSTANTS. 183
- 6.5 CREATE ARRAYS OF POINTERS. 186
- 6.6 BECOME ACQUAINTED WITH MULTIPLE INDIRECTION. 188
- 6.7 USE POINTERS AS PARAMETERS. 191

- 7. A Closer Look at Functions --- 195**
- 7.1 UNDERSTAND FUNCTION PROTOTYPES. 196
- 7.2 UNDERSTAND RECURSION. 207
- 7.3 TAKE A CLOSER LOOK AT PARAMETERS. 211
- 7.4 PASS ARGUMENTS TO main (). 215

- 7.5 COMPARE OLD-STYLE TO MODERN FUNCTION PARAMETER DECLARATIONS. 202

- 8. Console I/O --- 227**
 - 8.1 LEARN ANOTHER PREPROCESSOR DIRECTIVE. 229
 - 8.2 EXAMINE CHARACTER AND STRING INPUT AND OUTPUT. 233
 - 8.3 EXAMINE CHARACTER AND STRING INPUT AND OUTPUT. 233
 - 8.4 TAKE A CLOSER LOOK AT gets () AND puts (). 246
 - 8.5 MASTER printf (). 241
 - 8.6 MASTER scanf (). 246

- 9. FILE I/O --- 257**
 - 9.1 UNDERSTAND STREAMS. 259
 - 9.2 MASTER FILE-SYSTEM BASICS. 260
 - 9.3 UNDERSTAND FEOF () AND FERROR (). 269
 - 9.4 LEARN TO READ AND WRITE BINARY DATA. 278
 - 9.5 LEARN TO READ AND WRITE BINARY DATA 278
 - 9.6 UNDERSTAND RANDOM ACCESS. 285
 - 9.7 LEARN ABOUT VARIOUS FILE-SYSTEM FUNCTIONS. 290
 - 9.8 LEARN ABOUT THE STANDARD STREAMS. 293

- 10. STRUCTURES AND Unions -- 299**
 - 10.1 STRUCTURES AND UNIONS --- 299
 - 10.2 DECLARE POINTERS TO STRUCTURES. 314
 - 10.3 WORK WITH NESTED STRUCTURES. 318
 - 10.4 UNDERSTAND BIT-FIELDS. 324
 - 10.5 CREATE UNIONS. 329

- 11. Advanced data types and Operations --- 337**
 - 11.1 USE THE STORAGE CLASS SPECIFIERS. 339
 - 11.2 USE THE ACCESS MODIFIERS. 349
 - 11.3 DEFINE ENUMERATIONS. 352
 - 11.4 UNDERSTAND typedef. 356
 - 11.5 USE C'S BITWISE OPERATIONS. 358
 - 11.6 MASTER THE SHIFT OPERATORS. 363
 - 11.7 UNDERSTAND THE ? OPERATOR. 365
 - 11.8 DO MORE WITH THE ASSIGNMENT OPERATOR. 367

- 11.9 UNDERSTAND THE COMMA OPERATOR, 370
- 11.10 KNOW THE PRECEDENCE SUMMARY, 372

- 12. The C Preprocessor and Some Advanced Topics --- 375**
 - 12.1 LEARN MORE ABOUT #define AND #include, 377
 - 12.2 UNDERSTAND CONDITIONAL COMPILATION, 381
 - 12.3 LEARN ABOUT #error, #undef, #line, AND #pragma, 388
 - 12.4 EXAMINE C'S BUILT-IN MACROS, 391
 - 12.5 USE THE # AND ## OPERATORS, 393
 - 12.6 UNDERSTAND FUNCTION POINTERS, 395
 - 12.7 MASTER DYNAMIC ALLOCATION, 402

- A Some Common C Library Functions --- 411**
 - A.1 STRING AND CHARACTER FUNCTIONS, 412
 - A.2 THE MATHEMATICS FUNCTIONS, 424
 - A.3 TIME MATHEMATICS FUNCTIONS, 424
 - A.4 DYNAMIC ALLOCATION, 440
 - A.5 MISCELLANEOUS FUNCTIONS, 444

- B C Keyword Summary --- 457**

- C Building a Windows Skeleton --- 469**
 - WHICH VERSION OF WINDOWS?
 - WINDOWS PROGRAMMING PERSPECTIVE, 470
 - HOW WINDOWS AND YOUR PROGRAM INTERACT, 475
 - WINDOWS IS MULTITASKING, 474
 - THE WIN32 API, 474
 - THE COMPONENTS OF A WINDOW, 475
 - SOME WINDOW FUNCTION, 489
 - A SHORT WORD ABOUT DEFINITION FILES, 490
 - NAMING CONVENTIONS, 490
 - NAMING CONVENTIONS, 490
 - TO LEARN MORE, 490

- D Answers --- 493**

Index --- 633

Preface

This book teaches you how to program in what is usually regarded as the world's most important professional programming language: C.

One reason for C's success and staying power is that programmers like it. C combines subtlety and elegance with raw power and flexibility. It is a structured language that does not confine. It is a high-performance language that does not constrain. C is also a language that puts you, the programmer, firmly in charge. C was created by a programmer for programmers. It is not the contrived product of a committee, but rather the outcome of programmers seeking a better programming language.

C is important for another reason. It is the gateway to the world's two other professional programming languages: C++ and Java. C++ is built upon C, and Java is built upon C++. Thus, C is at the foundation of all modern programming, and knowledge of C is fundamental to the successful creation of high-performance, high-quality software. Simply put, to be a professional programmer today means that you are, competent in C.

A Short History of C

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

For many years, the de facto standard for C was the one described in *The C Programming Language* by Brian Kernighan and Dennis Ritchie (Prentice-Hall,) 1978. However, as C grew in popularity, a committee was organized in 1983 to create an ANSI (American National Standards Institute) standard for C. The standardization process took six years (much longer than anyone reasonably expected). The ANSI C standard was finally adopted late in 1989 and

TEACH YOURSELF

C

The first copies become generally available in 1990. The standard was amended slightly in 1996. Today, virtually all C compilers comply with ANSI standard C and that is the version of C you will learn in this book. (That is, this book teaches ANSI standard C.)

C is often referred to as a middle-language. Before C there were basically two types of languages used to program computers. One is called assembly language, which is the symbolic representation of the actual machine instructions executed by the computer. Assembly language is a low level language because the programmer is working with (in symbolic form) the actual instructions that the computer will execute. Assembly language can be used to create very efficient programs, but it provides no built-in control structures or I/O functions. All such items must be manually constructed by the programmer. By contrast, a high-level language buffers the programmer from the computer. A high-level language typically supplies various control structures, input and output commands, and the like, which make programming easier and faster. However, the elements of a high-level language may not relate directly to the way that the computer ultimately carries out the program. This separation often causes programs written using a high-level language to be less efficient than those written in assembly language. Because many people find assembly language programming to be a tedious, difficult task, there was a need for a language that balanced ease-of-use with efficiency. Many programmers feel that C provides this balance. It successfully combines the structure of a high-level language with the power and efficiency of assembly language. Since it spans the gap between assembly language and High-level languages, it is called a middle-level language.

Initially, C was used primarily for creating systems software. Systems software consists of the programs that help run the computer. These include programs such as operating systems, compilers, and editors. However, as C gained in popularity, it began to be used for general purpose programming. Today, C is used by programmers for virtually any programming task. It is a language that has survived the test of time and proven itself to be as versatile as it is powerful.

Cvs. C++

Newcomers are sometimes confused about the differences between C and C++ and how they relate to each other. In short, C++ is an extended version of C that is designed to support object-oriented programming (jOOP). C++ contains and supports the entire C language in addition to a set of object-oriented extensions. (That is, C++ is a superset of c.) Because C++ is built upon the foundation of C, you cannot learn C++ without learning the basics of C. Therefore, if you think that you will someday move on to C++, your knowledge of C will not only be useful, it will be necessary.

About this book

This book is unique because it teaches you the C language by applying mastery learning. It does so by presenting one idea at a time, followed by numerous examples and exercises to help you thoroughly understand each topic. This approach ensures that you master each topic before moving on.

The material is presented sequentially. Therefore, you should work carefully through each chapter because each chapter assumes that you know the material presented in all preceding chapters.

This book teaches ANSI standard C. This ensures that your knowledge will be applicable to the widest range of C environments. This book also uses contemporary syntax and structure, which means that you will be learning the right way to write C programs from the very beginning.

Now This Book is Organized

This book is composed of 12 chapters and 4 appendices. Each chapter (except Chapter 1) begins with a Review Skills Check, which consists of questions and exercises covering the previous chapter's material. The chapters are divided into sections. Each section covers one topic. At the end of each section are examples followed by exercises that test your understanding of the topic. At the end of each chapter, you will find a Mastery Skills check, which checks your knowledge of the material in the chapter. Finally, a Cumulative Skills Check is

TEACH YOURSELF

C

Presented that tests how well you are integrating new material with that presented in earlier chapters.

What's New in the Third Edition

For the most part, the form and structure of this book are unchanged from the previous two editions. Since C is a stable, standardized language, there was no reason to make major revisions. The two significant changes are the inclusion of full function prototypes in all programs, beginning with Chapter 1. Since all compilers now support—indeed, nearly demand—function prototyping, initial coverage of this issue was moved to the beginning of the book. Another set of changes was prompted by the emerging dominance of 32-bit environments. This caused a number of examples to be rewritten so that they would work for both 16- and 32-bit programs. In some places, additional examples, expanded coverage, or more exercises can be found. Finally, a few changes were made simply to reflect modern coding styles.

Conventions Used in this Book

Whenever a part of a program (such as a variable name) is referenced in text, it will be shown using boldface. Whenever a generic name is referenced in text, it will be shown in italics.

Will a C++ Compiler Work with C Programs?

Today, most compilers can compile both C and C++ programs. In fact, it is common to see a compiler advertised as a “C/C++” compiler, or sometimes just as a “C++” compiler. However, any and all compilers that can compile C++ programs can also compile C programs. Therefore, if your compiler calls itself a C++ compiler, don't worry, it is also a full, ANSI-standard C compiler.

If You're Using windows

If you use Windows and if your goal is to write Windows-based programs, then you have chosen the right language to learn. C is the language for which windows programming was designed, and the language in which many Windows programs are written. However, you will not be able to write Windows programs immediately. Here is why.

Windows programs are much more complex to create than non-Windows programs. Even a minimal, do-nothing Windows program makes use of several sophisticated C techniques, such as structures, pointers, and advanced function parameter types. For this reason, it is not possible to teach C programming by writing Windows programs because a Windows program assumes that you are an experienced C programmer!

If you are using Windows (rather than DOS, UNIX, etc.) on your computer, you can still learn C. However, you will need to run your programs from the DOS prompt because they will not be Windows based programs. All modern C C++ compilers will automatically create the correct environment to execute the programs shown in this Book, so it isn't something that you will typically need to worry about. If you will be writing Windows programs, you will want to read Appendix C after you complete this book. It contains a Windows skeleton program that you can try. Examining this skeleton will also show you that a thorough knowledge of C is required before windows programs can be written.

Don't forget:

Remember, the source code for all of the programs in this book is available free-of-charge on the Web at <http://www.osborne.com>. Downloading this code prevents you from having to type in the examples.

For Further Study

Teach yourself C, Third Edition is your gateway into the "Herb Schildt" series of programming books. Here is a partial list of Schildt's other programming books published by Osborne/McGraw-Hill.

If you want to learn more about C, you will find these books especially helpful.

C: The Complete Reference The Annotated ANSI C Standard

If you will be moving on to C++ (C's object-oriented extension), then you will find the Schildt's C++ books provide excellent coverage of this important language. We recommend

Teach Yourself C++
C++: The Complete Reference
C++ from the Ground Up

If you will be developing programs for the Web, you will want to read

Java: The Complete Reference

Co-authored by Herbert Schildt and Patrick Naughton.

Finally, if you want to program for Windows, we recommend

Schildt's Windows 95 Programming in C and C++
Schildt's Windows 95 Programming in C and C++
Windows NT 4 from the Ground Up
MFC programming from the Ground Up

When you need solid answer, fast, turn to
Herbert Schildt, the recognized authority on
programming.

