Robert Lafore

The Waite Group's® C Programming Using Turbo C++

Second Edition

Best-Seller Since 1987, Now Revised & Updated

Expert tips and programming advice

Ideal guide to C programming

Hands-on examples for using Turbo C++ or Borland C++

Disk features source code-for all the programming examples in the book

SAMS PUBLISHING

The Waite Group's

PROGRAMMING USING TURBO C®++

SECOND EDITION

Robert Lafore

SAMS PUBLISHING

A Division of Prentice Hall Computer Publishing
201 West 103rd Street, Indianapolis, IN 46290

This book is dedicated to Mitch Waite.

International Standard Book Number: 0-672-30399-X

Library of Congress Catalog Card Number: 93-86410

99 98 10 9

Interpretation of the printing code: the rightmost double-digit number is the year of the book's printing; the rightmost single-digit, the number of the book's printing. For example, a printing code of 93-1 shows that the first printing of the book occurred in 1993.

Composed in AGaramond and MCPdigital by Prentice Hall Computer Publishing

Printed in the United States of America

Trademarks

The Waite Group's C Programming Using Turbo C®++

Second Edition

From The Waite Group, Inc.

Editorial Director: *Scott Calamar*
Managing Editor: *John Crudo*
Technical Editors: *Harry Henderson, Joel Powell*

From Sams Publishing

Publisher: *Richard K. Swadley*
Acquisitions Manager: *Jordan Gold*
Development Editor: *Scott Palmer*
Editors: *Don MacLaren and Andy Saff*
Cover Designer: *Kathy Hanley*
Production: *Ayrika Bryant, Rich Evers, Mitzi Foster Gianakos, Dennis Clay Hager, Stephanie McComb, Angela M. Pozdol, Ryan Rader, and Dennis Wesner*
Indexers: *John Sleeva and Suzanne Snyder*

Foreword to the First Edition

by Ray Duncan

In just a decade, C has made the transition from an obscure Bell Laboratories house language to the programming language of choice for professionals. C has been used successfully for every type of programming problem imaginable—from operating systems to spreadsheets to expert systems—and efficient compilers are available for machines ranging in power from the Apple to the Cray. On the new wave of the windowing, graphics-intensive 68000-based personal computers such as the Macintosh, Amiga, and Atari, C is the *de facto* standard for serious developers.

Why has C met with such success when other languages, born in the same time-frame and designed with similar objectives, have gained only limited acceptance or have faded from view altogether? C's intimate association with UNIX—and the penetration into academia that was made possible by Bell Laboratory's broadminded licensing plans for universities—must be partly responsible, but the largest measure of C's success seems to be based on purely practical considerations: the portability of the compiler; the standard library concept; a powerful and varied repertoire of operators; a spare, elegant syntax; ready access to the hardware when needed; and the ease with which applications can be optimized by hand-coding isolated procedures.

The increasing dominance of C for both systems and applications programming has created a tremendous market for C books. I must confess that I am a compulsive buyer of computer books of every sort: an addiction based on curiosity, on a desire to learn from the approaches and styles of other authors, and on a continual search for excellent new books to recommend to readers of my column in *Dr. Dobb's Journal.* While indulging this pleasant but somewhat expensive habit over the last few years, I have built up quite a collection of C tutorials and reference works, but I have found nearly all of the introductory C books to be unsatisfying. The authors have difficulty discussing C without blurring the issues with UNIX-specific details, or they succumb to a fascination with clever C tricks, or they simply progress so quickly from elementary concepts to lengthy, complex example programs that they leave the reader behind.

When Mitchell Waite asked me if I would like to review Robert Lafore's manuscript for this book, I was surprised but gratified. I have long considered Robert's *Assembly*

Language Primer for the IBM PC and XT (New American Library, 1985) to set the standard of quality in its category, and I hoped that his C primer would be equally well organized, lucid, and complete. I certainly was not disappointed! I believe the book you are holding in your hands is the most accessible book on C that has yet been published. The material is presented in Robert's usual clear, forthright style, and the pace is steady but not intimidating. Best of all, the example programs are short but interesting, clearly demonstrate the concepts under discussion, and they are relevant to everyday programming problems on the IBM PC—you don't need to speak UNIX or buy your own VAX to reap their benefits.

Ray Duncan is a software developer and columnist for *Dr. Dobb's Journal* and the author of many computer books.

About the Author

Robert Lafore has been involved in the computer industry since 1965, when he learned assembly language on the DEC PDP-5, whose 4K of main memory was considered luxurious. He holds degrees in mathematics and electrical engineering; founded Interactive Fiction, a computer game company; and has served as managing editor of The Waite Group. Mr. Lafore is author of the best-selling *Assembly Language Primer for the IBM PC and XT, Microsoft C Programming for the PC, Object-Oriented Programming in Turbo C++,* and *Lafore's Window's Programming Made Easy,* among other titles. Mr. Lafore has also been a petroleum engineer in Southeast Asia, a systems analyst at the Lawrence Berkeley Laboratory, and has sailed his own boat to the South Pacific.

Preface

The first edition of this book was well received by readers, selling over 200,000 copies since its debut in 1987. Revisions over the years have kept it more or less up to date, (incorporating, for example, the transition from Kernighan and Ritchie C to ANSI C). However, despite these efforts, the book was beginning to show its age, and the author and publisher recognized that it was time for a complete housecleaning. Thus, the appearance of this Second Edition. Here are some highlights:

- All the example programs have been rewritten to compile with no errors or warnings with the latest version of the Turbo C++ compiler.
- A new chapter includes a working software application: an address-book database program that allows the user to store and retrieve names, addresses, and phone numbers. This program uses a modern Graphic User Interface (GUI).
- The second half of the book has been reorganized to give a more progressive presentation from general-to-specific and from easy-to-more-advanced material.
- Hardware-oriented sections have been collected from various places and combined into Chapter 15, "Hardware-Oriented C."
- A new chapter on the important subject of memory management has been created by combining considerable new material with existing text.

Of course, we've also endeavored to retain throughout the easy-to-read style that made the first edition so popular.

Acknowledgments for the Second Edition

My thanks to John Crudo and Scott Calamar of the Waite Group for their support and patience in the preparation of this new edition, to Harry Henderson for his usual eagle-eyed editing job, and to Joel Powell for his expertise in the technical edit.

Acknowledgments for the First Edition

I'd like to thank Mitchell Waite of The Waite Group for his dedicated and painstaking editing (complete with humorous asides and subliminal suggestions) and Harry Chesley, co-author of *Supercharging C* (Addison-Wesley), for his expert advice.

I am also indebted to the following individuals at Sams: Jim Hill for having the faith to buy this book idea, Damon Davis for publishing it, Jim Rounds for nursing it into the system, and Wendy Ford and Don Herrington for overseeing its production.

I would also like to thank Doug Adams for his technical edit, Bruce Webster of Byte magazine, Ray Duncan, and Herbert Schildt for reviewing the final manuscript, Harry Henderson for checking the final manuscript, and Don MacLaren of BooksCraft for his painstaking and skilled copy editing. In addition, thanks to Allen Holub for his editing of the revised edition.

Introduction

This book has two interconnected goals: to teach the C programming language and to show how C can be used to write applications for PC computers. That is, we not only explain C, but show how it's used in a real-world programming environment.

Borland's Turbo C++ for DOS (or the similar Borland C++) compiler provides the best platform yet devised for learning C. Its Integrated Development Environment (IDE) combines all the features needed to develop a C program—editor, compiler, linker, debugger, and help system—into a single, easy-to-use screen display. This enables you to develop programs far more simply and conveniently than you can using the traditional command-line compiler system. Turbo C++ is also very fast and surprisingly inexpensive.

Incidentally, don't be confused by the product name. Borland calls it *Turbo C++*, but it is really an extension of the old Turbo C, with C++ capabilities added. You can use this product (or Borland C++) for either C or C++ programming. In this book we use it for C. You don't need to know anything about C++.

With this book and Turbo C++ (or Borland C++), you'll be taking a fast and efficient path to mastery of the C language and the ability to write applications for PC computers.

What's Different About This Book?

Many introductory books on C present the language in a theoretical context, ignoring the machine the language is running on. This book takes a different approach: it teaches C in the context of PC (MS-DOS based) computers. There are several advantages to this approach.

First, focusing on a specific computer platform makes learning the language easier and more interesting. For example, graphics programming isn't standardized across platforms, so a generic C text can't use graphics. Programming examples must therefore concentrate on simple text-based interaction. In the PC world, on the other hand, we can make use of such features as graphics characters, program control of the cursor, and bit-mapped color graphics—capabilities that can enliven program examples and demonstrations.

Also, as we move to more complex aspects of the language, we can explain Turbo C++ constructs that might otherwise seem theoretical and mysterious—such as pointers and unions—by relating them to actual applications on the IBM hardware.

Finally, if your goal is to write programs specifically for PC computers, learning C in the PC environment—with examples that address specific aspects of PC hardware—will give you a head start in creating your own applications.

Learn by Doing

This book uses a hands-on approach. We believe that trying things out yourself is one of the best ways to learn, so examples are presented in the form of complete working programs, which can be typed and executed (or loaded from the accompanying disk). In general, we show what the output from the examples looks like, so you can be sure your program is functioning correctly. Very short examples are used at the beginning of the book, working toward longer and more sophisticated programs toward the end.

Illustrations and Exercises

We have also tried to make this a very visual book. Many programming concepts are best explained with pictures, so wherever possible we use figures to support the text.

Each chapter ends with a set of questions to test your general understanding of the material covered and programming exercises to ensure that you understand how to put the concepts to work. Answers to both questions and exercises are provided in the back of the book.

Why Use C?

The C programming language has become the overwhelming choice of serious programmers. Why? C is unique among programming languages in that it provides the convenience of a higher level language such as BASIC or Pascal, but at the same time allows much closer control of a computer's hardware and peripherals, as assembly language does. Most operations performed on the PC in assembly language can be accomplished—usually far more conveniently—in C. This is probably the principal reason for C's popularity on the PC.

C has other advantages, however. The better C compilers can now generate amazingly fast code. This code is so efficient that it's often difficult to produce significant speed increases by rewriting it in assembly language. C is also a well structured language: its syntax makes it easy to write programs that are modular and therefore easy to understand and maintain. The C language includes many features that are specifically designed to help create large or complex programs. Finally, C is portable: it is easier to convert a C program to run on a different machine than it is to convert programs written in most other languages.

C and C++

The C++ language is a superset of C: it adds object-oriented capabilities to the C language. C++ is most useful for large programming projects involving many programmers and many thousands of lines of code. Smaller programs are often easier to write in C. Also, C++ requires many new and difficult programming concepts. Unless you have a specific need for an object-oriented language, we suggest you stick with C, as this book does.

What You Should Know Before You Read This Book

No prior programming experience is necessary for you to use this book. C probably isn't quite as easy to learn as BASIC, but there's no reason a student can't plunge directly into C as a first programming language. Of course if you're already familiar with BASIC, Pascal, or some other language, that won't hurt.

You should be familiar with the MS-DOS operating system. You should be able to list directories and create, execute, copy, and erase files. You should also be familiar with tree-structured directories and how to move about in them.

What Equipment You Need to Use This Book

You can use this book with a variety of different hardware and software configurations. Here's what you'll need.

Hardware

You'll need a computer that runs the MS-DOS operating system. It should have at least an 80286 processor, but the more advanced the processor, the faster your programs will compile. You also need a hard (fixed) disk drive and a floppy drive. You must have a full 640K of conventional memory, plus at least 1MB of extended memory. (Practically, this means your computer must have 2MB or more of RAM.)

You can use a character-only monochrome display for most of the book; the exceptions being Chapter 10, "Turbo C++ Graphics," and part of Chapter 15, "Hardware-Oriented C," which deal with graphics. However, you'll be happier with a VGA color system. The graphics examples are written for VGA (and the Turbo C IDE looks better in color anyway).

A mouse is convenient (and fun) for operating Turbo C's IDE, which features pull-down menus, resizable windows, and other features of a typical graphics user interface. However, you can do everything from the keyboard if you don't have a mouse.

Although it isn't absolutely essential, you'll probably want a printer to generate program listings and record program output.

Software

You should use the MS-DOS operating system with a version number of 3.31 or greater. Of course, later versions are preferable.

If you're using Windows, you'll be interested to know that Turbo C++ works in Windows' MS-DOS compatibility box. All the example programs in this book also run in the DOS box.

You'll need one of two Borland compilers. The most usual choice is called *Turbo C++ for DOS.* It's widely available at computer stores and can also be ordered directly from Borland International, P.O Box 660001, Scotts Valley, CA 95067. (You can also use the older *Turbo C* product, Version 2.0 or greater.)

Another Borland product, Borland C++, is very similar to Turbo C++. Borland C++ may be sold separately, or bundled with Borland C++ and Application Frameworks, which contains a complete Windows development system. In either case, Borland C++ is considerably more expensive than Turbo C++, but if you already own it, you'll find it will work in almost exactly the same way as Turbo C++. We'll note the minor exceptions as we go along.

Turbo C++ (or Borland C++) is all you need to develop anything from a one-line program to a serious application with thousands of lines of code. There is no longer any necessity for a separate program editor or a separate debugger.

ANSI-Standard C

C was created by Dennis Ritchie at Bell Laboratories in the early 1970s. (C's predecessor was a language called *B*). In 1978, Brian Kernighan and Dennis Ritchie published a book, *The C Programming Language,* which for many years served as an effective definition of C.

In 1989, a new version of the language emerged from the American National Standards Institute: ANSI-standard C. This version includes many features (such as type `void` and function prototyping) that weren't present in the "traditional" Kernighan and Richie C. The Turbo C++ compiler, and this book, conform to the new ANSI-standard C, except where Turbo C++ provides useful additions to the standard.

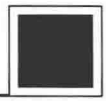
Typographical Conventions

In the text sections of this book (as opposed to program listings), all C-language keywords—such as `if`, `while`, and `switch`—are shown in a special monospace font to better distinguish them from the ordinary English-language usage of these words. Likewise, all C functions, including user-written functions and library functions such as `printf()` and `gets()`, are in mono, as are variable names.

When string constants such as `"Good morning"` and `"Fatal error"` are used in the text, they're set off by double quotes, just as they're in the C language itself. Character constants are set off by single quotes, again following the convention of the C language: `'a'` and `'b'` are characters.

Operators, which often consist of a single character such as (+) and (/), are often surrounded by parentheses for clarity. Keyboard keys, such as Ctrl key, Del key, and Z key, are capitalized as they are on the keyboard and followed by the word *key.*

Program names aren't in bold, but include their file extensions, such as .c in myprog.c and .obj in myprog.obj.



Good Luck, Lieutenant

This book covers a lot of ground—from simple one-line programs to complex graphics and database applications. Our intention is to make the exploration of this territory as easy and as interesting as possible, starting slowly and working up gradually to more challenging concepts. We hope we've succeeded and that you have as much fun reading this book as we had writing it.

Overview

Contents