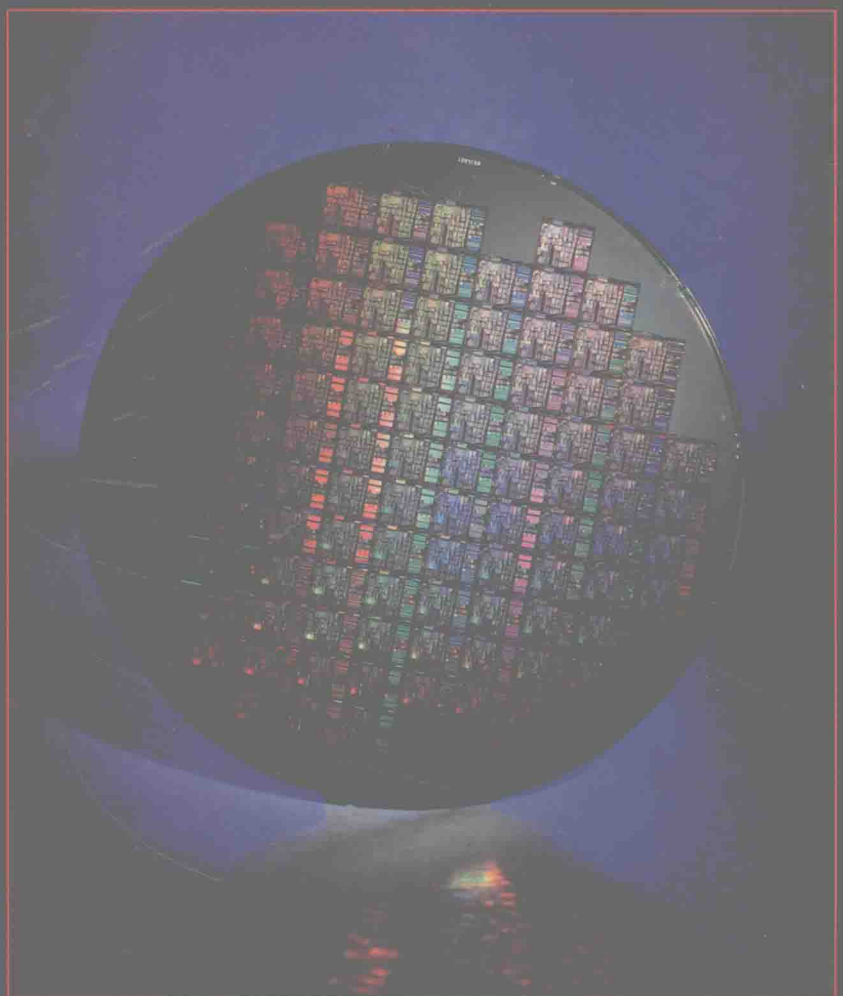
C for Scientists and Engineers



Richard Johnsonbaugh
Martin Kalin

C for and SCIENTISTS ENGINEERS

RICHARD JOHNSONBAUGH
DePaul University

MARTIN KALIN
DePaul University

Prentice Hall, Upper Saddle River, New Jersey 07458

Library of Congress Cataloging-in-Publication Data

Johnsonbaugh, Richard
C for scientists and engineers / Richard Johnsonbaugh, Martin Kalin.
p. cm.
Includes bibliographical references and index.
ISBN 0-02-361136-7
1. C (Computer program language). I. Kalin, Martin. II. Title
QA76.73.C15J655 1997
005.13'3--dc20 95-22669
CIP

Publisher: Alan Apt
Editor-in-Chief: Marcia Horton
Production Manager: Bayani Mendoza de Leon
Project Manager: Mona Pompili
Development Editor: Sondra Chavez
Managing Editor: Laura Steele
Copy Editor: Shirley Michaels

Design Director: Amy Rosen
Designer: Judy Matz-Coniglio
Illustrators: Patricia Guiterrez, Pete Ticola
Cover Designer: Heather Scott
Production Coordinator: Donna Sullivan
Editorial Assistant: Shirley McGuire

Printed in the United States of America

10 9 8 7 6 5 4 3 2 1

ISBN 0-02-361136-7

PRENTICE-HALL INTERNATIONAL (UK) LIMITED, *London*
PRENTICE-HALL OF AUSTRALIA PTY. LIMITED, *Sydney*
PRENTICE-HALL CANADA, INC., *Toronto*
PRENTICE-HALL HISPANOAMERICANA, S.A., *Mexico*
PRENTICE-HALL OF INDIA PRIVATE LIMITED, *New Delhi*
PRENTICE-HALL OF JAPAN, INC., *Tokyo*
SIMON & SCHUSTER ASIA PTE. LTD., *Singapore*
EDITORA PRENTICE-HALL DO BRASIL, LTDA., *Rio de Janeiro*

Preface

This book is intended for a one-term course in programming in C for scientists and engineers. We assume no prior programming knowledge. The book and its supplements—an *Instructor's Guide*, disk, and World Wide Web site—provide a comprehensive support system to help the reader master C. The book includes numerous examples, exercises, real world applications, programming exercises, lists of common programming errors, and figures.

Overview

C stands out among general-purpose programming languages for its unrivaled mix of portability, power, flexibility, and elegance. Because it compiles to highly efficient machine code, it is particularly well-suited to scientific and engineering applications. The C language presented in this book is based on standard C as developed and approved by the American National Standards Institute (ANSI) and the International Standards Organization (ISO).

This book includes the following features:

- An introductory chapter on algorithms, problem solving, computer systems, internal representation of data, programming languages, and program development. The chapter concludes with an explanation of the popularity of the C language and what it has to offer to scientific and engineering applications.
- Examples and exercises that cover a wide range of scientific and engineering applications.
- Real world applications.
- A broad variety of programming exercises. The book contains over 150 programming exercises. Answers to selected programming exercises are given in the *Instructor's Guide*.
- Sections on common programming errors.
- Discussion of standard C functions.

- Exercises at the ends of sections so that readers can check their mastery of the sections. The book contains over 650 such exercises. Answers to the odd-numbered section exercises are given in the back of the book, and answers to the even-numbered section exercises are given in the *Instructor's Guide*.
- Illustrations to facilitate the learning process. Two colors are used, not only to make the illustrations more attractive but also to differentiate input from output, to show the flow of control in the basic C constructs, and to highlight components of syntax diagrams.
- Helpful appendixes for reference and review.
- Code that compiles under both C and C++ compilers. Such code is known as "Clean C."
- Topics grouped according to their use and their relationships to one another. This organization enables readers to write simple but useful programs immediately and to skip or postpone some of the less often used and more esoteric parts of the language.
- Understandable code. When forced to choose between clarity and conciseness, we have opted for clarity.
- The preprocessor (Section 5.6).
- A thorough discussion of recursion (Section 5.8).
- Macros and header files used to write functions with an arbitrary number of arguments (Section 5.10).
- Full coverage of type qualifiers (Section 8.7).
- An introduction to linked lists, stacks, and queues (Sections 11.3 and 11.4).
- Assertions (Section 11.5).
- Exception handling and jumps (Section 11.6).
- Graphics support (Sections 11.7 and 11.8).
- A look ahead to C++ (Section 11.9).

Applications

In each chapter after the first, several sections are devoted to real world applications. Each of these sections contains a statement of a problem, sample input and output, a solution to the problem, and a well-documented implementation of the solution in C. Most of these sections conclude with an extended discussion. In some of the examples, these sections include a line-by-line discussion of the C program.

The **real world applications** include the following:

- Classifying solutions as acidic or nonacidic (Section 2.6)
- Statistics (Section 3.5)
- Bar graphs (Section 3.8)
- Summing a series (Section 4.8)
- Monte Carlo integration (Section 5.7)
- The Fourier transform (Section 6.3)
- Matrix multiplication (Sections 6.9 and 11.2)
- Solving a linear system of equations (Section 6.10)

- Sorting and searching (Sections 6.11 and 7.10)
- Forest fire percolation (Section 6.12)
- Interactive calculator (Section 8.4)
- Scheduling (Section 8.8)
- Random access files (Section 9.8)
- Complex numbers (Section 10.2)
- Chemical synthesis (Section 10.7)
- Fractals (Section 11.8)

Common Programming Errors

These sections highlight those aspects of the language that are easily misunderstood.

Examples

The book contains nearly **300** numbered examples, which clarify particular facets of C for the reader, show the purpose of various C features, and explain how to use C in various programming environments. A colored box (■) marks the end of each example.

Exercises

The book contains over **650** section review exercises, the answers to which are true or false, short answers, code fragments, and, in a few cases, entire programs. These exercises are suitable as homework problems or as self-tests. The answers to the odd-numbered exercises are given in the back of the book, and the answers to the even-numbered exercises are given in the *Instructor's Guide*. Class testing this book has convinced us of the importance of these exercises.

The applications covered in the programming exercises at the ends of the chapters include the following:

- Numerical solutions of differential equations (Programming Exercises 2.10, 2.11, and 5.24)
- Numerical approximation of derivatives (Programming Exercise 2.12)
- Graphing functions (Programming Exercises 3.5 and 3.6)
- Infinite series (Programming Exercises 4.4 and 7.22)
- Numerical integration (Programming Exercises 5.21 and 5.22)
- Root finding (Programming Exercise 5.23)
- Finding edges in a digital image (Programming Exercise 6.17)
- Game of *Life* (Programming Exercise 6.18)
- Matrix operations (Programming Exercise 6.21)
- Sorting (Programming Exercises 7.9 and 10.5)
- Complex numbers (Programming Exercise 10.3)
- Vector operations (Programming Exercise 10.4)
- Lagrange interpolation (Programming Exercise 10.6)

- Symbolic polynomial manipulation (Programming Exercises 11.9 and 11.10)
- Fractals (Programming Exercises 11.12 and 11.13)

Not every reader will be interested in all of these applications; however, we think that it is important to show the variety of scientific and engineering problems that C can address. Answers to selected programming exercises are given in the *Instructor's Guide*.

Organization of the Book

Chapter 1 serves as an introduction to computer systems and program development. (This chapter can be skipped if this material is already known.) We begin in Section 1.1 by discussing algorithms. Section 1.2 presents an overview of computer systems including hardware (main memory, CPU, etc.) and software (text editors, compilers, etc.). Internal representations of integers, floating-point numbers, characters, and instructions are discussed in Section 1.3. Section 1.4 continues with a discussion of programming languages (assembly languages and high-level languages). Section 1.5 summarizes the phases of program development: program specification, algorithm design, coding, and testing. In addition, several problem-solving strategies are given in Section 1.5. A discussion of the C language (Section 1.6) concludes Chapter 1.

Chapters 2 through 11 are devoted to C. Especially in the early chapters, we have grouped related C topics. For example, in Chapter 2 we discuss integer variables, the **`while`** loop, the **`do while`** loop, the **`if`** statement, and simple file handling. In this way, readers can immediately begin writing simple, but useful, programs. In Chapter 3, we discuss character, integer, and floating-point variables; arithmetic operators; relational and logical operators; the assignment operator in more detail; the **`for`** statement; and the increment and decrement operators. Less frequently used constructs such as the **`goto`** statement, labels, conditional expressions, and bitwise operators are discussed in Chapter 4. This organization contrasts with the organization of a manual in which one section is devoted to every data type available in C, another section is devoted to every C operator available, and so on. The sections that can be skipped or introduced later, as needed, have been marked with a dagger (†).

Chapter 5 deals with functions and program structure. Parameters and argument passing are covered in Sections 5.1 through 5.3. Section 5.6 treats the preprocessor in depth. Two sections are devoted to recursion. Section 5.8 discusses recursion in general and provides several short examples. Section 5.9 is a real world application that is solved recursively. Functions with an arbitrary number of arguments are discussed in Section 5.10.

Arrays and pointers (Chapters 6 and 7) follow Chapter 5 on functions. This order makes it possible to introduce interesting examples and programs early.

Storage classes are covered in Chapter 8. The discussion begins by assuming that the program resides in a single source file. (This is probably true of all the programs that the readers will have written to this point.) We then turn to storage classes in a program divided into two or more source files. In class testing this book, we have found this order of presentation to be the most successful. Section 8.7 discusses type qualifiers that are used to inform the compiler, especially an optimizing compiler, about what assumptions it can make about variables. Chapter 8 concludes with a real world application dealing with scheduling (Section 8.8).

The basic input/output functions for the standard input and standard output (e.g., **printf**, **scanf**), as well as the input/output functions for files (e.g., **fprintf**, **fscanf**), are introduced in Chapter 2. In Chapter 9, we treat these functions in depth and introduce additional input/output functions. Chapter 9 concludes with a real world application that shows how to implement a random access file in C using hashing.

Chapter 10 treats structures, unions, and enumerated types.

Chapter 11 contains advanced topics. Dynamic storage allocation is discussed and a real world application shows how to handle matrices using dynamic storage allocation. Linked lists, stacks, and queues are introduced. Assertions, exception handling and jumps, and graphics support for C are also covered. An *assertion* is a condition that must always be true at some particular point of the program's execution. Assertions are checked for correctness when the program is running. An *exception* is an unexpected error in a program. When an exception is detected in a program, the programmer can arrange for control to pass to a handler that attempts to deal with the exception. A real world application that draws a fractal is presented to show how to use a graphics package. Chapter 11 concludes with a brief introduction to the C++ (pronounced "C plus plus") programming language. C++ is an extension of C that supports object-oriented programming and also improves some parts of C.

This book is devoted to C independent of any particular operating system. However, the basic commands that one needs to compile, link, and run a C program are given in Appendix E for the UNIX system and in Appendix F for Borland C++ and Microsoft C++ under Windows and VAX-11 C under VAX/VMS. Borland C++ and Microsoft C++ contain both C++ and C compilers. In addition, redirection of input and output, so useful in C, is presented. Appendix E contains an extended discussion of C and UNIX.

We rely heavily on short examples, illustrations, tables, and other figures to illustrate specific points about C's syntax and semantics. From our own experience in teaching C and other languages, we are convinced that no single method is appropriate for clarifying every aspect about a language.

Most of our students agree with us that learning and using C is fun. We have tried to incorporate this view by using interesting examples, sample problems, programming exercises, and short slices of code.

Appendixes

Six appendixes are provided for reference. Appendix A contains ASCII and EBCDIC tables. Appendix B contains a summary of the C language, consisting of descriptions of the major constructs of C (e.g., **switch**, **while**) as well as a description of constants in C, the C data types, a summary of how functions work in C, initializing in definitions, a list of keywords, a summary of pointers in C, a table of the precedence of C operators, a list of the standard headers and their purposes, a summary of C storage classes, a summary of structures in C, and a summary of type qualifiers in C.

Complete two-color syntax diagrams of standard C may be found in Appendix C. Appendix D contains a list of some of the most useful C functions. We describe the parameters and return values for each function, the header file to include, and what the function does.

For those readers using C under UNIX, we have included Appendix E on UNIX and C. We discuss **cc** (the C compile command); the **man** (on-line help), **cb** (C beautifier), **grep**, **find**, and **make** utilities; the file system; pipes; directories and several commands

for navigating within directories (e.g., **pwd**, **mkdir**); commands for handling files (e.g., **ls**, **cp**); and run-time libraries.

Appendix F tells how to compile, link, and run a C program in VAX-11 C, Borland C++, and Microsoft C++. Explanations are included for single-file and multiple-file programs.

World Wide Web Site

The World Wide Web site http://condor.depaul.edu/~mkalin contains the source code, header files, and data files found on the program disk, sample syllabi, transparencies, and an errata list.

Disk

The IBM-format program disk, which is included in the *Instructor's Guide*, contains the source code, header files, and data files for all the book's real world applications, as well as the source code for some of the longer examples. Some programming exercises ask for modifications of these programs. In any case, we assume that many readers will want to experiment with the code that we provide on the disk.

Instructor's Guide

An *Instructor's Guide* is available at no cost to adopters of this book. The *Instructor's Guide* contains answers to the even-numbered section review exercises, transparency masters, sample syllabi, solutions to selected programming exercises, and the disk described previously.

Acknowledgments

We thank the following reviewers: Betty J. Barr, Avelino Gonzalez, Kathleen Kramer, and Thomas Walker.

We thank Hanyi Zhang for providing some of the solutions to programming exercises that appear in the *Instructor's Guide*.

We are indebted to the School of Computer Science, Telecommunications and Information Systems at DePaul University and its dean, Helmut Epp, for providing time and encouragement for the development of this book.

We received consistent support from the people at Prentice Hall. Special thanks for their help in preparing this edition go to Alan R. Apt, publisher; Mona Pompili, production editor; Shirley McGuire, editorial assistant; Sondra Chavez, developmental editor; and Laura Steele, managing editor.

Richard Johnsonbaugh
Martin Kalin

The basic input/output functions for the standard input and standard output (e.g., **printf**, **scanf**), as well as the input/output functions for files (e.g., **fprintf**, **fscanf**), are introduced in Chapter 2. In Chapter 9, we treat these functions in depth and introduce additional input/output functions. Chapter 9 concludes with a real world application that shows how to implement a random access file in C using hashing.

Chapter 10 treats structures, unions, and enumerated types.

Chapter 11 contains advanced topics. Dynamic storage allocation is discussed and a real world application shows how to handle matrices using dynamic storage allocation. Linked lists, stacks, and queues are introduced. Assertions, exception handling and jumps, and graphics support for C are also covered. An *assertion* is a condition that must always be true at some particular point of the program's execution. Assertions are checked for correctness when the program is running. An *exception* is an unexpected error in a program. When an exception is detected in a program, the programmer can arrange for control to pass to a handler that attempts to deal with the exception. A real world application that draws a fractal is presented to show how to use a graphics package. Chapter 11 concludes with a brief introduction to the C++ (pronounced "C plus plus") programming language. C++ is an extension of C that supports object-oriented programming and also improves some parts of C.

This book is devoted to C independent of any particular operating system. However, the basic commands that one needs to compile, link, and run a C program are given in Appendix E for the UNIX system and in Appendix F for Borland C++ and Microsoft C++ under Windows and VAX-11 C under VAX/VMS. Borland C++ and Microsoft C++ contain both C++ and C compilers. In addition, redirection of input and output, so useful in C, is presented. Appendix E contains an extended discussion of C and UNIX.

We rely heavily on short examples, illustrations, tables, and other figures to illustrate specific points about C's syntax and semantics. From our own experience in teaching C and other languages, we are convinced that no single method is appropriate for clarifying every aspect about a language.

Most of our students agree with us that learning and using C is fun. We have tried to incorporate this view by using interesting examples, sample problems, programming exercises, and short slices of code.

Appendixes

Six appendixes are provided for reference. Appendix A contains ASCII and EBCDIC tables. Appendix B contains a summary of the C language, consisting of descriptions of the major constructs of C (e.g., **switch**, **while**) as well as a description of constants in C, the C data types, a summary of how functions work in C, initializing in definitions, a list of keywords, a summary of pointers in C, a table of the precedence of C operators, a list of the standard headers and their purposes, a summary of C storage classes, a summary of structures in C, and a summary of type qualifiers in C.

Complete two-color syntax diagrams of standard C may be found in Appendix C. Appendix D contains a list of some of the most useful C functions. We describe the parameters and return values for each function, the header file to include, and what the function does.

For those readers using C under UNIX, we have included Appendix E on UNIX and C. We discuss **cc** (the C compile command); the **man** (on-line help), **cb** (C beautifier), **grep**, **find**, and **make** utilities; the file system; pipes; directories and several commands

Contents