



Problem Solving in C Including Breadth & Laboratories

Angela B. Shiflet



Problem Solving in C Including Breadth and Laboratories

江苏工业学院图书馆

Angela B. Shiflet

Wofford College

藏书章

with contributions from

Robert Martin

WEST PUBLISHING COMPANY

Minneapolis / St. Paul New York Los Angeles San Francisco

PRODUCTION CREDITS

Copyediting: Margaret C. Monahan

Design: John Rokusek

Composition: Carlisle Communications

Cover Image: FPG International Corp.

Photo Credits: **2**, Telegraph Colour Library/FPG International Corp.; **7 (top)**, Vannucci Foto-Services/FPG International Corp.; **7 (bottom)**, Victor Scocozza/FPG International Corp.; **12 (left & right)**, Courtesy of International Business Machines Corporation; **15 (top)**, ISU Photo Service; **15 (bottom)**, Courtesy of International Business Machines Corporation; **16 (top)**, Courtesy of International Business Machines Corporation; **16 (bottom)**, Historical Pictures/Stock Montage Inc.; **19**, Courtesy of International Business Machines Corporation; **21**, AT&T Archives; **30**, UPI/Bettmann; **43 (top)**, Freelance Photographer's Guild/FPG International Corp.; **43 (bottom left)**, Universal Pictures Shooting Star International; **43 (bottom right)**, S.S. Shooting Star; **58**, UPI/Bettmann; **80**, Photo courtesy of The Jet Propulsion Laboratory; **180**, Historical Pictures/Stock Montage; **342**, Ken Sherman/Phototake NYC; **399**, Historical Pictures/Stock Montage Inc.; **425**, Courtesy James E. Stoots, Jr., Lawrence Livermore National Laboratory. Run by The University of California for the Department of Energy. Printed with Permission of Cray Research, Inc.; **482**, Courtesy of International Business Machines Corporation; **506** Eric Kamp/Phototake NYC; **608**, UPI/Bettmann; **680**, Michael Freeman/Phototake NYC; **730**, Courtesy of International Business Machines Corporation; **909**, Courtesy of James E. Stoots, Lawrence Livermore National Laboratories.

WEST'S COMMITMENT TO THE ENVIRONMENT

In 1906, West Publishing Company began recycling materials left over from the production of books. This began a tradition of efficient and responsible use of resources. Today, up to 95 percent of our legal books and 70 percent of our college and school texts are printed on recycled, acid-free stock. West also recycles nearly 22 million pounds of scrap paper annually—the equivalent of 181,717 trees. Since the 1960s, West has devised ways to capture and recycle waste inks, solvents, oils, and vapors created in the printing process. We also recycle plastics of all kinds, wood, glass, corrugated cardboard, and batteries, and have eliminated the use of Styrofoam book packaging. We at West are proud of the longevity and the scope of our commitment to the environment.

Production, Prepress, Printing and Binding by West Publishing Company.



TEXT IS PRINTED ON 10% POST CONSUMER RECYCLED PAPER



British Library Cataloguing-in-Publication Data. A catalogue record for this book is available from the British Library.

COPYRIGHT © 1995

By WEST PUBLISHING COMPANY

610 Opperman Drive

P.O. Box 64526

St. Paul, MN 55164-0526

All rights reserved

Printed in the United States of America

02 01 00 99 98 97 96 95 8 7 6 5 4 3 2 1 0

Library of Congress Cataloging-in-Publication Data

Shiflet, Angela B.

Problem solving in C including breadth and laboratories / Angela

B. Shiflet.

p. cm.

Includes index.

ISBN 0-314-04554-6 (soft)

1. C (Computer program language) 2. Problem solving--Data processing. I. Title.

QA76.73.C15S474 1995

005.13'3--dc20

94-37379

CIP

**Dedicated to my husband,
George,
and my parents,
Isabell and Carroll Buzzett**



Preface

Problem Solving in C, Including Breadth and Laboratories introduces the beginning computer science student to the analysis, design, implementation, testing, and debugging of programs using ANSI C and to the breadth and richness of the computer science discipline. The text has a top-down approach to programming and presents material in a clear, visual manner with ample use of examples and figures. Instructors can tailor their courses in a variety of ways using this flexible text.

The student is introduced to functions in Chapter 1 and the *if* and *switch* statements in Chapter 3. This early coverage allows programs with some “meat” to be introduced fairly early in the term. Moreover, the separation of the discussions of integers (Chapter 2), floating point numbers (Chapter 4), and characters (Chapter 7) allows the student to focus on fewer concepts at a time. This organization holds the complexity to a minimum. The typical treatment of including all types in one chapter is like watering a lawn. After a while, the water simply runs off and is wasted. This text’s organization allows maximum absorption of the material and concepts. The early presentation of functions and the gradual introduction of data types and syntax allow programming principles and problem solving to evolve as the language constructs are developed. The text emphasizes problem solving throughout, with several sections focusing on this topic. Moreover, a clear, straight-forward presentation of all topics is included, with good separation between each.

Each chapter concludes with a laboratory section that is truly integrated with the topics in the text. The laboratories give a wonderful hands-on introduction to many features of problem solving with C. Students can work through one or all laboratory exercises in a self-paced fashion or in a closed laboratory environment. Each laboratory moves through the chapter concepts, easing a student into writing and debugging programs. For example, the laboratory for Chapter 3 teaches the student how to write programs with stubs. The student uses a program from the laboratory disk, which is simply a program with a set of stub functions. The laboratory gives the student directions for fleshing out and then testing the functions—one function at a time—exactly as it would be done in the real world. When the laboratory is complete, the student has a program that he or she has put together using stubs. Such a laboratory builds programming confidence. The laboratory breaks the work into simple tasks that ensure student success. Hands-on computer work makes a student more confident and adventurous with a language. Program templates in several laboratories promote good design. Moreover, experimentation is encouraged, and several laboratories in later chapters (Chapters 10, 11, 14, and 15) have a teamwork component. (*The Instructor’s Manual* has suggestions for individual assignments as alternatives to the team assignments.) Most laboratories have several exercises with multiple parts. Thus, to meet individual time requirements and needs, an instructor can assign one or all these exercises. Programming segments and data files to accompany the laboratories are on a disk included with the text. The instructor’s disk contains answers to laboratory exercises.

A professor can cover all or some of a variety of breadth sections. This material presents a broad range of topics from the discipline of computer science. For example, breadth material includes such topics as the object-oriented paradigm, intellectual property, invention of the first computers, logic, color in computer graphics, machine and assembler languages, external storage, formal grammars, memory, and databases. The text disk includes the source code for a CPU simulator program, which executes the example machine language of Section 12.7. A computer graphics package on the disk accompanies Section 11.7. The package contains files of device-dependent and independent routines and device drivers (files of device-dependent routines) for Turbo C and Think C. Moreover, an outline of a generic driver can be used to develop drivers for other systems. The Learning Features section below contains a complete list of the 26 breadth sections. These breadth sections enhance the subject material, help place topics in perspective, and give students a preview of the discipline of computer science. Students like to see the relevance of the subject and what is ahead for them, and these sections give a good taste of the future.

The style of writing in the breadth material, laboratories, and text material is clear, direct, and readable. Students love examples, and there are many in the text. Large “case studies,” as well as shorter examples, are developed in a top-down fashion and described with structure charts, pseudocode, and pre- and postconditions. Concrete examples make abstract concepts come alive.

Numerous figures accompany the examples and explanations. These figures help today’s visually oriented students to “see” what happens inside the computer as each instruction executes. A number of figures show the movement of data and the effects of certain instructions on storage locations. For example, Figure 2.15 of Section 2.9 follows each change in memory with each line of the program. Color highlights changes in the figures and important segments of code. This visual orientation of the code and figures is even more important as students move to more abstract ideas.

Figures and examples help to explain the material, but students learn by doing. Each section has a number of exercises that correlate directly to the material. Answers to problems with numbers in color are located in Appendix J. Some exercises—such as those related to searching and sorting—have the students perform the task by hand before coding it. This kind of drill makes abstract concepts more concrete. The exercises are complete and thorough and have a good mix of easy and challenging questions. The text also includes questions from Graduate Record Computer Science Examinations. The *Instructor’s Manual* contains answers to the remaining exercises. The C code in the text, the manual, and the disk has been computer tested.

Besides exercises, most sections contain programming projects, which range in difficulty and topics. These projects provide an additional source of applications. Some projects involve revising earlier projects, and several projects are from the Programming Contest sponsored by Fairleigh Dickinson University.

Along with programming projects and exercises, numerous features help students concentrate on important concepts. Each chapter begins with an introduction and list of goals. Programming and debugging hints at the end of each chapter cover such topics as walkthrough technique, clarity of user interface, debugging techniques, some errors C compilers do not flag, and mistaken operator symbols. Appendix H contains a summary of the UNIX dbx, Turbo C, and Think C debuggers.

The closing material of each chapter contains key terms, a summary, and review questions. The list of key terms includes page numbers, which make this feature useful for reviewing. The chapter summary helps to focus the reader on important points. Review questions and answers are a tremendous study aid.

Appendices include an ASCII table, keywords, operator precedence, conversion specifications, summary of file I/O, random number generators, contents of text disk, debugging on different systems, a Glossary, answers to selected exercises, and answers to review questions.

Learning Features

Breadth Material

At least one section in each chapter covers the breadth of computer science. These topics mesh with the chapter's material. The professor can cover all, some, or none of these topics. As the following list reveals, the 26 breadth sections complement the chapters in which they occur.

- 1 The Fundamentals of Computer Science
 - 1.2 The Discipline of Computer Science
 - 1.4 Invention of the First Computers
 - 1.6 The History of C
- 2 Integer Variables, Expressions, and Functions
 - 2.4 Storage of Integers in the Computer
 - 2.5 Integer Arithmetic in the Computer
- 3 Making Decisions
 - 3.5 Logic
- 4 Additional Numeric Types
 - 4.2 Storage of Floating Point Numbers
- 5 Looping
 - 5.6 Computer Time
 - 5.7 Truncation Error in Loops
- 6 Counter-Controlled Loops
 - 6.3 A Technique of Numerical Computing
 - 6.4 Intellectual Property
- 7 Characters
 - 7.4 Octal and Hexadecimal Number Systems
- 8 Arrays
 - 8.6 Color in Computer Graphics
- 9 Pointers
 - 9.2 Memory
- 10 Strings and String Functions
 - 10.7 Software Life Cycle for Large Systems
- 11 Structures and User-Defined Types
 - 11.3 Databases
 - 11.7 A Computer Graphics Package (accompanying software on text disk)
- 12 Levels of Programming Abstraction
 - 12.4 Some Operating System Features
 - 12.5 The Object-Oriented Paradigm
 - 12.6 C++: Object-Oriented Programming
 - 12.7 Machine and Assembler Languages (accompanying software on text disk)
- 13 Recursion
 - 13.3 Formal Grammars

- 14 Input/Output and Files
 - 14.2 Secondary Storage
- 15 Binary Operations
 - 15.2 Logic Gates
 - 15.3 Logic Circuits
- 16 Data Structures
 - 16.4 Run-Time Stack during Program Execution

Laboratories

Each chapter has a laboratory module with accompanying code on a disk. Some laboratories involve experimental methods. Others explore alternative implementations. All reinforce the material in the text. For example, the laboratory in the chapter on recursion has the student perform an experiment to compare the efficiency of the three summation algorithms, one nonrecursive and two recursive solutions. The following four laboratories employ the team approach:

- Chapter 10 Develop a command-driven, line-oriented text editor
- Chapter 11 Develop a stock portfolio program
- Chapter 14 Maintain a program
- Chapter 15 Formulate external documentation

The *Instructor's Manual* suggests variations for instructors who prefer individual to team assignments. Moreover, the instructor can use a laboratory in a scheduled, supervised environment or can assign parts of the laboratory for independent exploration by the student.

Example Operations and Applications

The text is example-driven. Most sections start with careful detailed discussions and simple examples to illustrate each new concept and end with a longer example illustrating analysis, design, and implementation. The level slowly increases as the reader progresses through the text. The organized approach to examples—particularly with accompanying diagrams—aids understanding of the subject.

Numerous Diagrams Highlighted with Color

Diagrams help students visualize the actions of operations and algorithms. Color emphasizes changes. For example, figures in Section 9.1 on The Concept of Pointers help to illuminate this challenging topic.

Section Exercises

Exercises appear at the end of each section, not just at the end of the chapter. These include short answer problems, diagrams of the execution of segments, design and coding of functions, applications, and questions from the Graduate Record Computer Science Examination. The text contains more than 1000 exercises in all.

Answers to Exercises

Answers to some exercises (those with numbers in color) appear in Appendix J which allows students to check their work for immediate reinforcement. The *Instructor's Manual* contains answers to the remaining exercises. Answers involving C code have been computer tested.

Programming Projects	An average of 15 programming projects are included per chapter. These major assignments allow students to design, code, and test. By completing such a project, the student enhances his or her understanding of the material and abilities in software development. For ease of assignment, projects are listed at the ends of the sections.
Historical Anecdotes	Such anecdotes add interest to the text and make computer science history more real. For example, Chapter 1's Programming and Debugging Hints contains the story of Grace Murray Hopper finding a "bug" in the computer. Moreover, the historical anecdotes often present material that a computer science major should know about the history of the discipline.
Chapter Introductions	An introduction at the beginning of each chapter gives an overview of the material in the chapter.
Chapter Goals	A list of study goals for the chapter follows the introduction.
Programming and Debugging Hints	Because students spend much time debugging programs, the hints sections are very useful.
Key Terms	Using the Key Terms section, students can test their knowledge of the important terms in the chapter. Because page numbers accompany the terms, students can readily check their answers or consult the text to refresh their memories.
Summary	The Summary presents a concise overview of chapter material.
Chapter Review Questions	For self-examination, each chapter also contains a list of review questions. Answers are in Appendix K.

Supplementary Materials

Instructor's Manual	An <i>Instructor's Manual</i> contains solutions to text exercises, answers to at least one project per chapter, additional test problems with answers, laboratory code answers, transparency masters, and suggestions for lectures. The accompanying disk has examples from the text, data files, laboratory exercises, and their answers. Code in the <i>Instructor's Manual</i> and on the disk have been computer tested.
Test Bank	A <i>Test Bank</i> on disk and in the <i>Instructor's Manual</i> contains test questions and answers for each chapter.

- Laboratory Manual** A *Laboratory Manual* with disk contains the chapter laboratories. The manual has additional room for a student's notes and answers.
- Text Disk** Included with the text is a disk of laboratory programs, program examples from the text, and data files in ANSI C.
- Overhead Transparency Masters** Transparency masters of key figures, algorithms, and programs are available in the *Instructor's Manual*.

Testing of Code

The source code appearing in this textbook, *Instructor's Manual*, and the accompanying diskette was prepared and tested on a Macintosh 840 AV using Symantec's THINK C or Symantec C++ compiler, Version 6.0, and on a Mitsuba 80386 MS-DOS PC using Borland's C++ compiler, Versions 3.1 and 4.0. Every effort was made to ensure ANSI compliance and thus provide the student with portable example programs and code fragments. In all cases, the target execution environment is MS-DOS. These programs are not designed to be compiled or executed as MS Windows applications. Although these programs can be compiled using a Windows-based compiler or integrated development environment—such as Borland's C++ for Windows—the student must correctly specify the target environment and run the resulting programs within an MS-DOS shell.

Acknowledgments

Any project of this magnitude requires the cooperation and support of many people. The author gratefully acknowledges the many friends, colleagues and students for their help in the completion of this work. For his ideas and contributions to programming and problem solving, thanks go to Robert Martin. William Campbell and Jason Womick have been of enormous help—William through the manuscript preparation, text disk production, and glossary compilation; Jason in generating solutions for the exercises. Christine Clawson helped in checking the art and compiling the index. Helen Thomas gave much proofreading assistance.

At West Publishing, Peter Gordon has been a wonderful editor, giving valuable direction, imagination and encouragement. Michelle McAnelly, the production editor, did a fantastic job orchestrating the production phase of the project. Thanks also go to Peggy Monohan for her accurate copy editing, and to John Rokusek for the attractive design.

It is impossible to thank adequately John Hinkel, my friend and former colleague at Lander University. Not only has John been a source of many valuable ideas and insights, but also he has provided tremendous encouragement and enthusiasm throughout this project.

I would also like to acknowledge the administration of Wofford College, particularly Dan Maultsby, who provided encouragement and a reduced teaching load to write this book.

Some of the programming problems were contributed by faculty, staff, and students of the Department of Mathematics and Computer Science at Fairleigh Dickinson University, Madison New Jersey. These problems were compiled from those used in the University's annual programming contest over the last eight years. Particular thanks go to Dr. Peter Falley, Dr. Phil Laplante, and Ralph Knapp.

GRE test questions were selected from *The Graduate Record Examinations Descriptive Booklet 1991–93*, 1991 and *Practicing to Take the GRE Computer Science Test, 2nd Edition*, 1992, Educational Testing Service. Reprinted by permission of Educational Testing Service. Permission to reprint GRE materials does not constitute review or endorsement by Educational Testing Service of this publication as a whole or of any other testing information it may contain.

Borland Corp. contributed copies of Turbo C and Symantec Corp. donated Symantec C++ for use in the project.

I am grateful to the following reviewers who offered many valuable constructive criticisms:

John Lowther
Michigan Technological University
E. Terry Magel
Kentucky State University
Matthew Dickerson
Middlebury College
Ronald A. Mann
University of Louisville
Bill Stockwell
University of Central Oklahoma
Marguerite K. Summers
Sangamon State University
Sharon Underwood
Livingston University
Sanjay Jain
National University of Singapore
(previously of University of Delaware)
Paul Morneau
Adirondack Community College
Robert Geitz
Oberlin College
Stephen P. Leach
Florida State University
Grace Anne Crowder
Towson State University
Peg Eaton
Plymouth State College
Lorraine Callahan
Northern Arizona University
Jeffrey A. Slomka
Southwest Texas State University
Tim Davis
University of Florida, Gainesville

Margaret Anne Pierce
Georgia Southern University
Ronald J. Gould
Emory University
Neil R. Sorensen
Weber State University
John Carroll
San Diego State University
Nathaniel G. Martin
University of Rochester
Mike Michaelson
Palomar College
Reggie Kwan
Montana College of Mineral Science and Technology
James M. Frazier
University of North Carolina at Charlotte
Stephen J. Allan
Utah State University
A. M. Fayek
California State University, Chico
Brian Malloy
Clemson University
Richard J. Botting
California State University, San Bernardino
Peter J. Gingo
University of Akron
Dwayne A. McCalister
California State University, Fresno
Marty J. Wolf
Mankato State University
Susan M. Simons
Memphis State University

My husband, George W. Shiflet, Jr., has encouraged me throughout this project and has done so much to make it possible for me to have the time to write. George and my parents, Isabell and Carroll Buzzett, have given me boundless love and support. It is to these three wonderful people that I dedicate this book.



Contents



1 The Fundamentals of Computer Science, 1

Introduction, 1

- 1.1 Solving Problems with the Computer, 2
 - An Overview of Problem Solving, 3
 - Analyzing the Problem, 4
 - Designing a Solution, 5
 - Implementing the Design, 8
 - Testing the Code, 8
 - Maintaining the Product, 8
 - Summary, 8
 - Section 1.1 Exercises, 9
- 1.2 Breadth: The Discipline of Computer Science, 9
 - Theory Paradigm, 10
 - Abstraction Paradigm, 10
 - Design Paradigm, 11
- 1.3 Model of a Computer System, 11
 - Input and Output Devices, 11
 - Secondary Storage, 11
 - Central Processing Unit, 13
 - Main Memory, 14
 - Section 1.3 Exercises, 14
- 1.4 Breadth: Invention of the First Computers, 15
 - Section 1.4 Exercises, 17
- 1.5 Steps to Execution, 17
 - Editor, 17
 - Preprocessor, 17
 - Compiler, 18
 - Linker, 20
 - Section 1.5 Exercises, 21
- 1.6 Breadth: The History of C, 21
- 1.7 Implementation of the Design, 22
 - A Program to Display a Message, 23
 - Comments, 23
 - Inclusion of *stdio.h*, 24
 - main*, 25
 - printf*, 25
 - Semicolon, 26
 - Style, 26
 - Section 1.7 Exercises, 28
 - Section 1.7 Programming Projects, 29

1.8	Top-Down Design and Functions, 30
	Using Library Functions, 30
	Connecting Functions to Top-Down Design, 31
	Function Definition, 33
	Calling a Function, 34
	Function Prototype, 35
	ANSI C Libraries, 36
	Section 1.8 Exercises, 39
	Section 1.8 Programming Projects, 39
1.9	Breadth: Subject Areas of Computer Science, 40
	Algorithms and Data Structures, 41
	Architecture, 41
	Artificial Intelligence and Robotics, 42
	Database and Information Retrieval, 42
	Human-Computer Communication, 42
	Numerical and Symbolic Computation, 44
	Operating Systems, 44
	Programming Languages, 44
	Software Methodology and Engineering, 45
	Social, Ethical, and Professional Issues, 45
	Programming and Debugging Hints, 46
	Debugging, 46
	Walkthrough Technique, 47
	Modular Programming, 47
	Key Terms, 47
	Summary, 48
	Review Questions, 50
	Laboratory, 51



2 Integer Variables, Expressions, and Functions, 55

Introduction, 55

2.1	Integer Data, 56
	Variables, 57
	Variable Declaration, 57
	Naming of Variables, 59
	Section 2.1 Exercises, 60
2.2	The Assignment Statement, 60
	Lvalues and Rvalues, 62
	Labeled Output, 65
	Not an Algebraic Formula, 66
	Declaration-Initialization, 67
	Section 2.2 Exercises, 67
	Section 2.2 Programming Projects, 68
2.3	Integer Arithmetic, 68
	Four Binary Operators, 68
	Modulus Operator, 71
	Printing %, 74
	Unary Minus, 75

	Operator Precedence, 75
	Section 2.3 Exercises, 77
	Section 2.3 Programming Projects, 78
2.4	Breadth: Storage of Integers in the Computer, 79
	Binary Representation of Integers, 79
	Counting, 81
	Decrementing by 1, 82
	Range of Unsigned Integers in a Computer, 82
	Conversion of a Decimal Integer to Binary, 83
	Section 2.4 Exercises, 85
	Section 2.4 Programming Projects, 86
2.5	Breadth: Integer Arithmetic in the Computer, 86
	Signed-Magnitude Representation, 86
	Two's Complement Representation, 87
	Addition, 89
	Subtraction, 91
	Multiplication and Division by Two, 91
	Section 2.5 Exercises, 92
	Section 2.5 Programming Project, 93
2.6	Interactive Programs, 93
	Interactive versus Batch Programs, 93
	Interactive Programs in C, 94
	Section 2.6 Exercises, 96
	Section 2.6 Programming Projects, 96
2.7	Problem Solving with Integer Functions, 97
	Preconditions and Postconditions, 97
	Analysis and Design of a Function, 99
	Implementation of an Integer Function, 100
	Procedures, 101
	Arguments and Parameters, 104
	Default Type Declarations for Functions, 110
	Section 2.7 Exercises, 111
	Section 2.7 Programming Projects, 112
2.8	Problem Solving Revisited, 112
	Analysis, 113
	Section 2.8 Exercises, 114
2.9	Scope of Variables, 115
	Local Variables and Scope, 115
	Pass by Value, 115
	Local Variables with the Same Name, 118
	Global Variables, 120
	Section 2.9 Exercises, 123
	Programming and Debugging Hints, 125
	Clarity of Comments, 125
	Clarity of Code, 126
	Clarity of User Interface, 126
	Key Terms, 127
	Summary, 127
	Review Questions, 129
	Laboratory, 130



3 Making Decisions, 137

Introduction, 137

- 3.1 Relational and Logical Operators, 138
 - Relational Operators, 139
 - Logical Operators, 140
 - Boolean Constants, Expressions, and Variables, 142
 - Operator Precedence, 143
 - Section 3.1 Exercises, 145
- 3.2 Selection, 145
 - Flow of Control, 145
 - The *if* Statement, 146
 - The *if-else* Statement, 148
 - Section 3.2 Exercises, 155
 - Section 3.2 Programming Projects, 156
- 3.3 Nesting, 157
 - Section 3.3 Exercises, 166
 - Section 3.3 Programming Projects, 167
- 3.4 Multiple-Way Selection, 167
 - The *switch* Statement, 167
 - Branching to the Same Point, 176
 - Section 3.4 Exercises, 177
 - Section 3.4 Programming Projects, 179
- 3.5 Breadth: Logic, 180
 - George Boole and Edmund Berkeley, 180
 - Basic Components of Logic, 180
 - Truth Tables, 181
 - Algebra of Propositions, 183
 - DeMorgan's Laws, 184
 - Section 3.5 Exercises, 184
- 3.6 Testing Schemes, 185
 - Top-Down Testing, 186
 - Bottom-Up Testing, 190
 - Combined Top-Down and Bottom-Up Testing, 191
 - Section 3.6 Exercises, 192

Programming and Debugging Hints, 192

- Decision Control Structures, 192
- Testing, 194

Key Terms, 194

Summary, 195

Review Questions, 197

Laboratory, 198



4 Additional Numeric Types, 205

Introduction, 205

- 4.1 Floating Point Numbers, 206
 - Distinctions between Integers and Floating Point Numbers, 206
 - Floating Point Arithmetic, 207

Exponential Notation, 208
Printing Numbers, 209
Type <i>double</i> , 212
Section 4.1 Exercises, 213
Section 4.1 Programming Projects, 213
4.2 Breadth: Storage of Floating Point Numbers, 214
Conversion from Base 2 to Base 10, 214
Conversion from Base 10 to Base 2, 214
Multiplication and Division by 2, 216
Storage of Floating Point Numbers, 216
Truncation Error, 218
Section 4.2 Exercises, 220
4.3 Coercion, 221
Implicit Coercion, 221
Explicit Coercion, 222
Strong and Weak Typing, 224
Section 4.3 Exercises, 225
Section 4.3 Programming Projects, 226
4.4 Additional Integer Types, 226
Different Sizes of Integers, 226
Unsigned Integers, 227
Mixed-Mode Arithmetic, 227
Section 4.4 Exercises, 228
4.5 ANSI C Header Files and <i>#define</i> , 229
Numerical Constants, 229
Defining Preprocessor Constants, 231
Absolute Value Function, 232
Square Root Function, 232
Additional Math Library Functions, 233
Section 4.5 Exercises, 242
Section 4.5 Programming Projects, 243
Programming and Debugging Hints, 244
Interfaces between Functions: Global Variables, 244
Preprocessor Constants, 245
Reader's Understanding of the Interface, 245
Key Terms, 246
Summary, 246
Review Questions, 247
Laboratory, 249



5 Looping, 257

Introduction, 257

- 5.1 Updating Assignment Operators, 258
 - Increment and Decrement Operators, 259
 - Pre- and Post-increment and Decrement, 260
 - Section 5.1 Exercises, 263
- 5.2 Looping with a Pretest, 264
 - The *while* Loop, 264
 - Infinite Loop, 268

- Nature of the Pretest, 269
- Manipulation of Loop Variable, 270
- Section 5.2 Exercises, 276
- Section 5.2 Programming Project, 277
- 5.3 Looping with a Posttest, 277
 - The *do-while* Loop, 277
 - Applications, 278
 - Section 5.3 Exercises, 286
 - Section 5.3 Programming Projects, 287
- 5.4 Looping and Interactive Programs, 287
 - The Sentinel Technique, 287
 - Random Numbers in Interactive Programs, 290
 - Seeding the Random Number Generator, 291
 - Ranges of Random Numbers, 292
 - Section 5.4 Exercises, 300
 - Section 5.4 Programming Projects, 301
- 5.5 Structured Programming, 302
- 5.6 Breadth: Computer Time, 303
 - Clock Cycle, 304
 - Clock Frequency, 305
 - Section 5.6 Exercises, 306
- 5.7 Breadth: Truncation Error in Loops, 307
 - Section 5.7 Exercises, 309
- Programming and Debugging Hints, 310**
 - Updating Assignment Operators, 310
 - Assignment and Relational Equals Operators, 311
- Key Terms, 312**
- Summary, 312**
- Review Questions, 313**
- Laboratory, 314**



6 Counter-Controlled Loops, 317

- Introduction, 317**
- 6.1 The *for* Loop, 318
 - Loop Choice, 320
 - Counting Down, 320
 - Tables, 321
 - Section 6.1 Exercises, 325
 - Section 6.1 Programming Projects, 326
- 6.2 Nesting of Loops, 328
 - Section 6.2 Exercises, 340
 - Section 6.2 Programming Projects, 341
- 6.3 Breadth: A Technique of Numerical Computing, 342
 - Section 6.3 Programming Projects, 348
- 6.4 Breadth: Intellectual Property, 349
 - Copyright Law, 349
 - Patents, 350
 - The Company Perspective, 351
 - Section 6.4 Exercises, 352