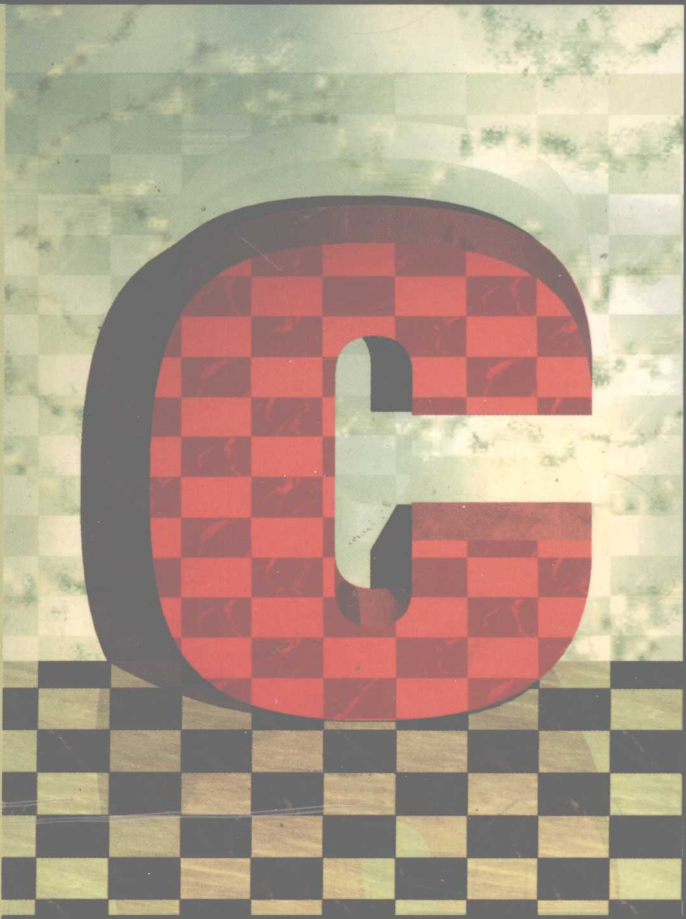

PROBLEM SOLVING AND PROGRAM DESIGN IN C

second edition

Hanly · Koffman

PROBLEM SOLVING AND PROGRAM DESIGN IN C

second edition

Jeri R. Hanly
UNIVERSITY OF WYOMING

Elliot B. Koffman
TEMPLE UNIVERSITY

Addison-Wesley Publishing Company
Reading, Massachusetts • Menlo Park, California • New York • Don Mills, Ontario
Wokingham, England • Amsterdam • Bonn • Sydney • Singapore • Tokyo • Madrid
San Juan • Milan • Paris

Lynne Doran Cote *Sponsoring Editor*
Helen M. Wythe *Senior Production Editor*
Maite Suarez-Rivas and Anita Devine *Assistant Editors*
Lisa Delgado *Text Designer*
Tech-Graphics *Illustrations*
Eileen Hoff *Cover Design Supervisor*
Diana Coe *Cover Design*
Roy Logan *Senior Manufacturing Manager*
Michael and Sigrid Wile *Composition*

Library of Congress Cataloging-in-Publication Data
Hanly, Jeri R.
Problem solving and program design in C / Jeri R. Hanly, Elliot B. Koffman. -- 2nd ed.
p. cm.
Includes index.
ISBN 0-201-59063-8
1. C (Computer program language) I. Koffman, Elliot B. II. Title.
QA76.73.C15H363 1996
005.13'3--dc20 95-7320
CIP

3 4 5 6 7 8 9 10–DOC–989796

PROBLEM SOLVING AND PROGRAM DESIGN IN C

second edition

To our families
Brian, Eric, and
Kevin Hanly;
Caryn, Richard,
Deborah, and
Robin Koffman

This textbook teaches a disciplined approach to solving problems and to applying widely accepted software engineering methods to design program solutions as cohesive, readable, reusable modules. We present as an implementation vehicle for these modules a subset of ANSI C—a standardized, industrial-strength programming language known for its power and portability. This text can be used for a first course in programming methods: It assumes no prior knowledge of computers or programming. The text's broad selection of case studies and exercises allows an instructor to design an introductory programming course in C for computer science majors or for students from a wide range of other disciplines.

In preparing this new edition, we have made every effort to simplify the presentation wherever possible, striving for more focused chapters. For example, we have consolidated the coverage of relational, equality, and logical operators in Chapter 4 where conditions are first needed for selection structures. We have placed all three loop constructs in Chapter 5 along with the operators with side effects (++, +=, and so on).

This edition discusses user-defined functions with input parameters earlier than did Edition 1, presenting these functions in Chapter 3 and using them regularly thereafter. Besides expanding our coverage of ANSI C string library functions, we have added coverage of enumerated types, functions as parameters, stacks, and dynamic memory allocation. We also present several data structures composed of dynamically allocated nodes, including linked lists (used as stacks, as queues, and as ordered lists) and binary trees.

An important style change in this new edition is the early introduction of function prototypes. Consequently, our standard source file format in this edition is (1) preprocessor directives, (2) function prototypes, (3) function `main`, and (4) other function definitions.

Using C to Teach Program Development

Two of our goals—teaching program design and teaching C—may be seen by some as contradictory. C is widely perceived as a language to be tackled only after one has learned the fundamentals of programming in some other, friendlier language. The perception that C is excessively difficult is traceable to the history of the language. Designed as a vehicle for programming the UNIX operating system, C found its original clientele among programmers who understood the complexities of the operating system and the underlying machine, and who considered it natural to exploit this knowledge in their programs. Therefore, it is not surprising that many textbooks whose primary goal is to teach C expose the student to program examples requiring an understanding of machine concepts that are not in the syllabus of a standard introductory programming course.

In this text we are able to teach both a rational approach to program development and an introduction to ANSI C because we have chosen the first goal as our primary one. One might fear that this choice would lead to a watered-down treatment of ANSI C. On the contrary, we find that the blended presentation of programming concepts and of the implementation of these concepts in C captures a focused picture of the power of ANSI C as a high-level programming language, a picture that is often blurred in texts whose foremost objective is the coverage of all of ANSI C. Even following this approach of giving program design precedence over discussion of C language features, we have arrived at a coverage of the essential constructs of C that is quite comprehensive.

Pointers and the Organization of the Book

The order in which C language topics are presented is dictated by our view of the needs of the beginning programmer rather than by the structure of the C programming language. The reader may be surprised to discover that there is no chapter entitled "Pointers." This missing chapter title follows from our treatment of C as a high-level language, not from a lack of awareness of the critical role of pointers in C.

Whereas other high-level languages have separate language constructs for output parameters and arrays, C openly folds these concepts into its notion of a pointer, drastically increasing the complexity of learning the language. We simplify the learning process by discussing pointers from these separate perspectives where such topics normally arise when teaching other programming languages, thus allowing a student to absorb the intricacies of pointer usage a little at a time. Our approach makes possible the presentation of fundamental concepts using traditional high-level language terminology—output parameter, array, array subscript, string—and makes it easier for students without prior assembly language background to master the many facets of pointer usage.

Therefore, this text has not one, but four chapters that focus on pointers. Chapter 6 discusses the use of pointers as simple output and input/output parameters, Chapter 8 deals with arrays, Chapter 9 presents strings and arrays of pointers, and Chapter 14 describes dynamic memory allocation. In addition, Chapters 2 and 12 discuss file pointers.

New to this Edition: Applications Written in C

A new feature of this edition is a collection of brief articles presenting applications written in C. Included are descriptions of Vivo320, a video-conferencing tool; LINEUP, a database system for criminal mug shots; and the Borland C/C++ compiler. In addition, one article traces the history of the joint development of UNIX and C.

Software Engineering Concepts

The book presents many aspects of software engineering. Some are explicitly discussed and others are taught only by example. The connection between good problem-solving skills and effective software development is established early in Chapter 1 with a section that discusses the art and science of problem solving. The five-phase software development method presented in Chapter 1 is used to solve the first case study and is applied uniformly to case studies throughout the text. Major program style issues are highlighted in special displays, and the coding style used in examples is based on guidelines followed in segments of the C software industry. There are sections in several chapters that discuss algorithm tracing, program debugging, and testing.

Chapter 3 introduces procedural abstraction through selected C library functions, parameterless void functions, and functions that take input parameters and return a value. Chapters 4 and 5 include additional function examples, and Chapter 6 completes the study of functions that have simple parameters. The chapter discusses the use of pointers to represent output and input/output parameters, and Chapter 7 introduces the use of a function as a parameter.

Case studies and sample programs in Chapters 6, 8, and 11 introduce by example the concepts of data abstraction and of encapsulation of a data type and operators. Chapter 13 presents C's facilities for formalizing procedural and data abstraction in personal libraries defined by separate header and implementation files.

The use of visible function interfaces is emphasized throughout the text. We do not mention the possibility of using a global variable until Chapter 13, and then we carefully describe both the dangers and the value of global variable usage.

Pedagogical Features

We employ several pedagogical features to enhance the usefulness of this book as a teaching tool. Some of these features are discussed below.

End-of-Section Exercises Most sections end with a number of self-check exercises. These include exercises that require analysis of program fragments as well as short programming exercises. Answers to selected self-check exercises appear at the back of the book; answers to the rest of the exercises are provided in the instructor's manual.

End-of-Chapter Exercises A set of quick-check exercises with answers follows each Chapter Review. There are also review exercises whose solutions appear in the instructor's manual.

End-of-Chapter Projects Each chapter ends with a set of programming projects. Answers to selected projects appear in the instructor's manual.

Examples and Case Studies The book contains a wide variety of programming examples. Whenever possible, examples contain complete programs or functions rather than incomplete program fragments. Each chapter contains one or more substantial case studies that are solved following the software development method. Numerous case studies give the student glimpses of important applications of computing, including database searching, business applications such as billing and sales analysis, word processing, environmental applications such as radiation level monitoring and water conservation.

Syntax Display Boxes The syntax displays describe the syntax and semantics of new C features and provide examples.

Program Style Displays The program style displays discuss major issues of good programming style.

Error Discussions and Chapter Review Each chapter concludes with a section that discusses common programming errors. A chapter review includes a table of new C constructs.

Appendixes and Supplement

A reference table of ANSI C constructs appears on the inside covers of the book, and the first appendix presents character set tables. Because this text covers only a subset of ANSI C, the remaining appendixes play an especially vital role in increasing the value of the book as a reference. Appendix B is an alphabetized table of ANSI C library facilities. Appendix C gives a table showing the precedence and associativity of all ANSI C operators; the operators not previously defined are explained in this appendix. Throughout the book, array referencing is done with subscript notation; Appendix D is the only coverage of pointer arithmetic. Appendix E lists all ANSI C reserved words.

Source Code

An on-line version of the source code figures is available at our anonymous ftp site. To access, set your ftp to `aw.com`. At the prompt, log in as `anonymous` and use your internet address as the password. From there, you change to the directory `cd cseng/authors/hanly/cs1.2e`.

Instructor's Manual

The Instructor's Manual includes chapter by chapter summaries and suggestions based on selected textbook figures. These are available via the `aw.com` ftp site. You will need to contact your sales rep for the password. Please see directions above under "Source Code" on how to access this site.

Solutions and Test Questions

Test questions and solutions to the internal self check, review questions and selected programming projects are available by contacting your local Addison-Wesley sales representative.

Acknowledgments

Many people participated in the development of this book. We thank especially Cindy Johnson, who developed the articles on C applications, and Paul W. Abrahams, Kenneth Pugh of Pugh-Killeen Associates, Oliver Jones of Vivo Software Inc., and Michael R. Weisert of Borland International Inc., who provided the material for these articles. We thank Joan C. Horvath of the Jet Propulsion Laboratory, California Institute of Technology, for contributing several new programming exercises. We are grateful for the work of several Temple University and University of Wyoming students and former students who helped to verify the programming examples and who provided answer keys for the host of exercises. These include Mark Thoney, Lynne Doherty, Andrew Wrobel, Steve Babiak, Donna Chrupcala, Masoud Kermani, and Thayne Routh.

The principal reviewers were enormously helpful in suggesting improvements and in finding errors. They include: Steve Allan, Utah State University; Michael Beeson, San Jose State - Chico; John Lewis, Villanova University; James Schmolze, Tufts University; Larry Sells, Oklahoma City University; Travis Tull, University of Tulsa; Richard Weinand, Wayne State University; and Beth Weiss, University of Arizona.

It has been a pleasure to work with the Addison-Wesley team in this endeavor. The sponsoring editor, Lynne Doran Cote, provided much guidance and encouragement throughout all phases of manuscript revision. Her assistants, Maite Suarez-Rivas and Anita Devine, coordinated the review process and the preparation of the instructor's manual and handled a great variety of other details. Helen Wythe supervised the design and production of the book, while Tom Ziolkowski developed the marketing campaign.

J.R.H.
E.B.K.

1. Overview of Computers and Programming 1

1.1 Electronic Computers Then and Now 2
1.2 Introduction to Computer Hardware 5
1.3 Overview of Programming Languages 12
1.4 Processing a High-Level Language Program 14
1.5 The Software Development Method 17
1.6 Applying the Software Development Method 21
Case Study: Converting Miles to Kilometers 21
Chapter Review 24

2. Overview of C 27

2.1 C Language Elements 28
2.2 Variable Declarations and Data Types 35
2.3 Executable Statements 39
2.4 General Form of a C Program 49
2.5 Arithmetic Expressions 53
Case Study: Evaluating a Collection of Coins 61
2.6 Formatting Numbers in Program Output 67
2.7 Interactive Mode, Batch Mode, and Data Files 70
2.8 Common Programming Errors 75
Chapter Review 81

3. Top-Down Design with Functions 89

3.1 Building Programs from Existing Information 90
Case Study: Finding the Area and Circumference of a Circle 91
Case Study: Computing the Weight of a Batch of Flat Washers 94
3.2 Library Functions 99
3.3 Top-Down Design and Structure Charts 106
Case Study: Drawing Simple Diagrams 106

3.4 Functions without Arguments 108
3.5 Functions with Input Arguments 119
3.6 Common Programming Errors 130
Chapter Review 130

4. Selection Structures: if and switch Statements 139

4.1 Control Structures 140
4.2 Conditions 140
4.3 The if Statement 152
4.4 If Statements with Compound Statements 156
4.5 Decision Steps in Algorithms 160
Case Study: Water Bill Problem 160
4.6 More Problem Solving 170
Case Study: Water Bill with Conservation Requirements 171
4.7 Nested if Statements and Multiple-Alternative Decisions 173
4.8 The switch Statement 184
4.9 Common Programming Errors 190
Chapter Review 192

5. Repetition and Loop Statements 203

5.1 Repetition in Programs and the while Statement 204
5.2 Computing a Sum or Product in a Loop 208
5.3 Counting Loops and the for Statement 214
5.4 Conditional Loops 224
5.5 Loop Design 230
5.6 Nested Loops 238
5.7 The do-while Statement and Flag-Controlled Loops 242
5.8 Problem Solving Illustrated 247
Case Study: Computing Radiation Levels 247
5.9 How to Debug and Test Programs 252
5.10 Common Programming Errors 255
Chapter Review 258

6. Modular Programming 269

6.1 Functions with Simple Output Parameters 270
6.2 Multiple Calls to a Function with Input/Output Parameters 279
6.3 Scope of Names 285
6.4 Formal Output Parameters as Actual Arguments 287
6.5 A Program with Multiple Functions 291
Case Study: Arithmetic with Common Fractions 292
6.6 Debugging and Testing a Program System 302
6.7 Common Programming Errors 305
Chapter Review 306

7. Simple Data Types 315

7.1 Representation and Conversion of Numeric Types 316
7.2 Representation and Conversion of type char 323
7.3 Enumerated Types 326
7.4 Iterative Approximations 332
Case Study: Bisection Method for Finding Roots 334
7.5 Common Programming Errors 342
Chapter Review 343

8. Arrays 355

8.1 Declaring and Referencing Arrays 356
8.2 Array Subscripts 360
8.3 Using for Loops for Sequential Access 362
8.4 Using Array Elements as Function Arguments 368
8.5 Array Arguments 371
8.6 Searching and Sorting an Array 385
8.7 Multidimensional Arrays 390
8.8 Array Processing Illustrated 396
Case Study: Analysis of Sales Data 397
8.9 Common Programming Errors 405
Chapter Review 407

9. Strings 417

9.1 String Basics 418
9.2 String Library Functions: Assignment and Substrings 424
9.3 Longer Strings: Concatenation and Whole-Line Input 433
9.4 String Comparison 438
9.5 Arrays of Pointers 441
9.6 Character Operations 447
9.7 String-to-Number and Number-to-String Conversions 453
9.8 String Processing Illustrated 460
Case Study: Text Editor 460
9.9 Common Programming Errors 469
Chapter Review 471

10. Recursion 481

10.1 The Nature of Recursion 482
10.2 Tracing a Recursive Function 487
10.3 Recursive Mathematical Functions 495
10.4 Recursive Functions with Array and String Parameters 501
Case Study: Finding Capital Letters in a String 502
Case Study: Recursive Selection Sort 505
10.5 Problem Solving with Recursion 508
Case Study: Operations on Sets 509
10.6 A Classic Case Study in Recursion: Towers of Hanoi 517
10.7 Common Programming Errors 522
Chapter Review 524

11. Structure and Union Types 531

11.1 User-Defined Structure Types 532
11.2 Structure Type Data as Input and Output Parameters 538
11.3 Functions Whose Result Values Are Structured 544

11.4 Problem Solving with Structure Types 547
Case Study: A User-Defined Type for Complex Numbers 548
11.5 Parallel Arrays and Arrays of Structures 556
Case Study: Universal Measurement Conversion 559
11.6 Union Types (Optional) 568
11.7 Common Programming Errors 575
Chapter Review 575

12. Text and Binary File Processing **585**

12.1 Input/Output Files: Review and Further Study 586
12.2 Binary Files 594
12.3 Searching a Database 601
Case Study: Database Inquiry 601
12.4 Common Programming Errors 611
Chapter Review 612

13. Programming in the Large **621**

13.1 Using Abstraction to Manage Complexity 622
13.2 Personal Libraries: Header Files 625
13.3 Personal Libraries: Implementation Files 630
13.4 Storage Classes 633
13.5 Modifying Functions for Inclusion in a Library 639
13.6 Conditional Compilation 642
13.7 Arguments to Function main 647
13.8 Defining Macros with Parameters 649
13.9 Common Programming Errors 654
Chapter Review 655

14. Dynamic Data Structures **663**

14.1 Pointers and the Functions malloc and calloc 664
14.2 Linked Lists 671
14.3 Linked List Operators 677
14.4 Representing a Stack with a Linked List 683
14.5 Representing a Queue with a Linked List 687

14.6 Ordered Lists 693
Case Study: Maintaining an Ordered List of Integers 694
14.7 Binary Trees 705
14.8 Common Programming Errors 716
Chapter Review 716

Appendixes

A Character Sets AP1
B ANSI C Standard Libraries AP3
C C Operators AP21
D Pointer Arithmetic AP27
E ANSI C Reserved Words AP29

Answers A1

Index I1

Overview of Computers and Programming

chapter 1

1.1 Electronic Computers Then and Now

1.2 Introduction to Computer Hardware

1.3 Overview of Programming Languages

1.4 Processing a High-Level Language

1.5 The Software Development Method

1.6 Applying the Software Development Method
Case Study: Converting Miles to Kilometers

Chapter Review