THIRD EDITION UPDATE

Problem Solving & Program Design in C

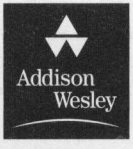
Jeri R. Hanly
Elliot B. Koffman

PROBLEM SOLVING AND PROGRAM DESIGN IN C

third edition update

Jeri R. Hanly
UNIVERSITY OF WYOMING

Elliot B. Koffman
TEMPLE UNIVERSITY



Boston San Francisco New York
London Toronto Sydney Tokyo Singapore Madrid
Mexico City Munich Paris Cape Town Hong Kong Montreal

Executive Editor: Susan Hartman Sullivan
Editorial Assistant: Galia Shokry
Composition: Michael and Sigrid Wile
Copyeditor: Stephanie Magean
Proofreader: Elizabeth Bailey
Cover Designer: Leslie Haimes
Manufacturing and Prepress: Caroline Fell

Access the latest information about Addison-Wesley books from our World Wide Web site:
http://www.aw.com/cs

Figures 1.7, 1.8, and 1.11 are reprinted from *Essential C++ for Engineers and Scientists,* (figs. 1.5, 1.6, and 1.9), by Jeri Hanly.

Library of Congress Cataloging-in-Publication Data

Hanly, Jeri R.
Problem solving and program design in C / Jeri R. Hanly, Elliot B. Koffman.— 3rd ed. update.
p. cm.
ISBN 0-201-75490-8 (pbk.)
1. C (Computer program language) I. Koffman, Elliot B. II. Title.

QA76.73.C15 H363 2002
005.13'—dc21

2001041320
CIP

ISBN 0-201-75490-8

1 2 3 4 5 6 7 8 9 10-CRW-030201

This book is dedicated to our families.

Jeri Hanly's family:
Brian, Kevin, and Trinity
Eric and Jennifier Hanly

Elliot Koffman's family:
Caryn and Debbie
Robin and Jeff
Richard, Jacquie, and Dustin Koffman

PREFACE

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 edition, we have added Chapter 15, which can serve as a transition to the study of C++ in a subsequent course. We have expanded the first section of our chapter on iteration (Chapter 5) so that we introduce the full range of loops conceptually before delving into their C implementations, and we have included in Chapter 12 more extensive coverage of escape sequences and format specifiers used with `printf`. In addition, we have modified Chapter 14 so that it helps students consolidate their understanding of pointers as arrays, output parameters, and file accessors just prior to their exploration of the role of the pointer in dynamic memory allocation.

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 after reviewing pointer uses previously covered. In addition, Chapters 2 and 12 discuss file pointers.

Applications Written in C

This text illustrates the importance of the C programming language by including 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. Chapter 15 introduces the concept of object-oriented design as implemented by C++.

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 the following pedagogical features to enhance the usefulness of this book as a teaching tool:

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.

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 applica-

tions 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.

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.

Appendixes, CD-ROM, and Supplements

Appendix F describes how to use Borland C++ Builder version 5 (see below). It also describes how to use the free command-line interpreter which can be downloaded from the Borland website (www.Borland.com). A reference table of ANSI C constructs appears on the inside covers of the book, and Appendix A 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 standard libraries. 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.

CD-ROM with Borland C++ Builder 5

The textbook comes with a CD-ROM that contains Borland C++ Builder 5. C++ Builder is an Integrated Development Environment for creating, debugging, and running C and C++ programs.

Source Code

An on-line version of the source code figures is available at our anonymous ftp site. To access, set your ftp to `ftp.awl.com`. At the prompt, log in as `anonymous` and use your internet address as the password. From there, you change to the directory `cseng/authors/hanly/cs1.3e`.

Instructor's Manual

The Instructor's Manual includes chapter by chapter summaries and suggestions based on selected textbook figures. These are available online. You will need to contact your sales rep for the password and access instructions.

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 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. We also thank Jeri's Computer Science Department colleagues at the University of Wyoming who have been so willing to answer her questions—Allyson Anderson, Mark Arnold, and Robin Hill.

It has been a pleasure to work with the Addison-Wesley team in this endeavor. The sponsoring editor, Susan Hartman, along with her assistant, Galia Shokry, provided much guidance and encouragement throughout all phases of manuscript revision. Patty Mahtani supervised the production of the book, while Michael Hirsch developed the marketing campaign.

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

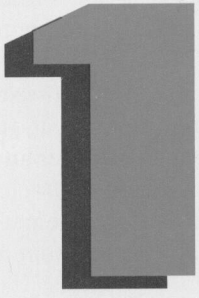
CONTENTS

6. Modular Programming 277

7. Simple Data Types 323

8. Arrays 363

Overview of Computers and Programming

chapter 

computer
a machine that can receive, store, transform, and output data of all kinds

In developed countries, life at the end of the twentieth century is conducted in a veritable sea of computers. From the coffeepot that turns itself on to brew your morning coffee to the microwave that cooks your breakfast to the automobile that you drive to work to the automated teller machine you stop by for cash, virtually every aspect of your life depends on **computers**. These machines that receive, store, process, and output information can deal with data of all kinds: numbers, text, images, graphics, and sound, to name a few.

The computer program's role in this technology is essential; without a list of instructions to follow, the computer is virtually useless. Programming languages allow us to write those programs and thus to communicate with computers.

You are about to begin the study of computer science using one of the most versatile programming languages available today: the C language. This chapter introduces you to the computer and its components and to the major categories of programming languages. It discusses how C programs are processed by a computer. It also describes a systematic approach to solving programming problems called the software development method and shows you how to apply it.

1.1 ELECTRONIC COMPUTERS THEN AND NOW

In our everyday life, we come in contact with computers frequently, some of us using computers for word processing or even having studied programming in high school. But it wasn't always this way. Not so long ago, most people considered computers to be mysterious devices whose secrets were known only by a few computer wizards.

The first electronic computer was built in the late 1930s by Dr. John Atanasoff and Clifford Berry at Iowa State University. Atanasoff designed his computer to assist graduate students in nuclear physics with their mathematical computations.

The first large-scale, general-purpose electronic digital computer, called the ENIAC, was completed in 1946 at the University of Pennsylvania with funding from the U.S. Army. Weighing 30 tons and occupying a 30-by-50-foot space, the ENIAC was used to compute ballistics tables, predict the weather, and make atomic energy calculations.

These early computers used vacuum tubes as their basic electronic component. Technological advances in the design and manufacture of electronic components led to new generations of computers that were considerably smaller, faster, and less expensive than previous ones.

computer chip (microprocessor chip)
a silicon chip containing the circuitry for a computer processor

Using today's technology, the entire circuitry of a computer processor can be packaged in a single electronic component called a **computer** or **microprocessor chip** (Fig. 1.1), which is about the size of a postage stamp. Their afford-