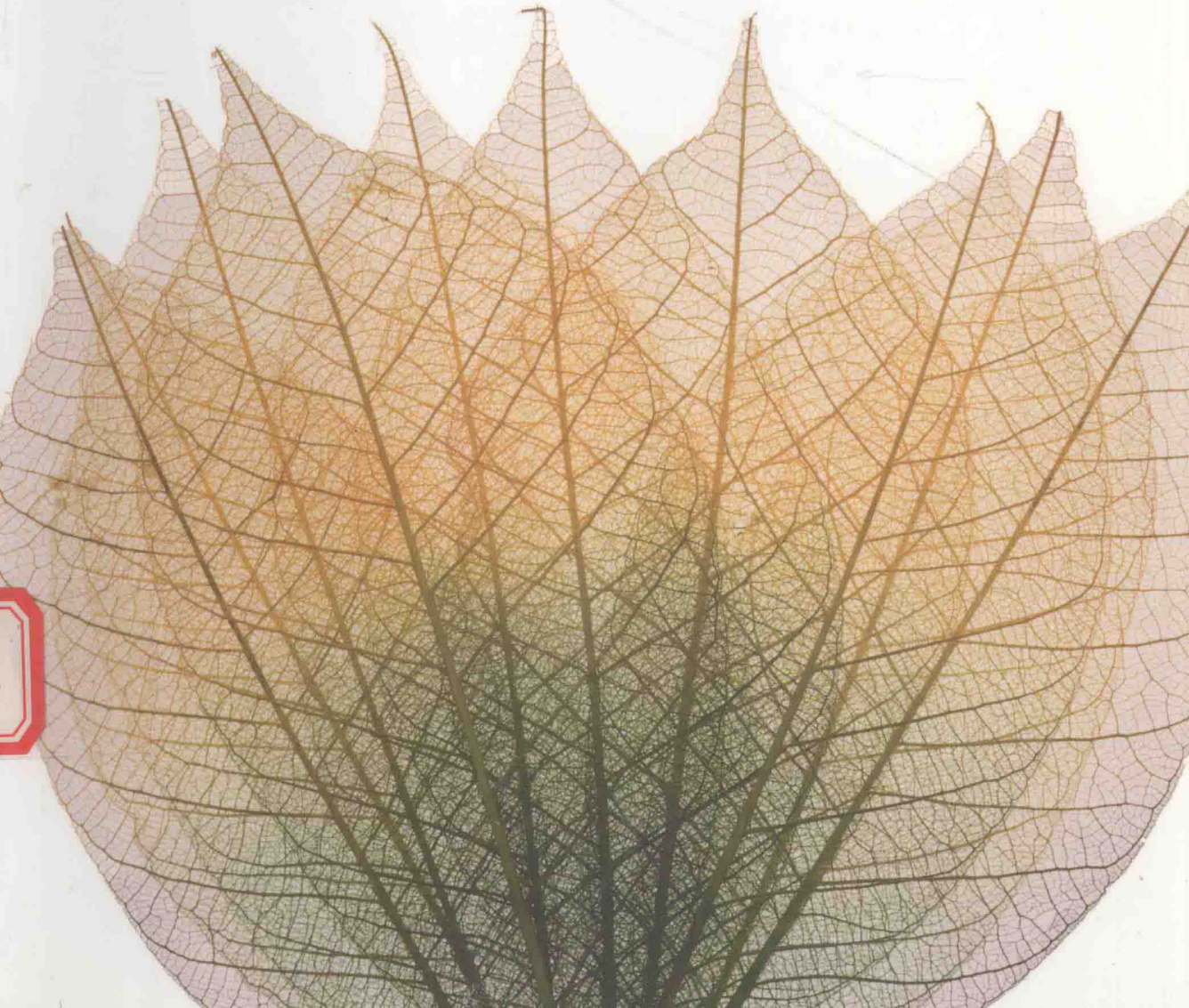
Eighth Edition

PROGRAMMING LOGIC AND DESIGN

Comprehensive

Joyce Farrell



EIGHTH EDITION

PROGRAMMING LOGIC AND DESIGN

COMPREHENSIVE VERSION

JOYCE FARRELL

CENGAGE Learning

Australia • Brazil • Japan • Korea • Mexico • Singapore • Spain • United Kingdom • United States

Programming Logic and Design, Comprehensive version, Eighth Edition
Joyce Farrell

Senior Product Manager: Jim Gish

Senior Content Developer: Alyssa Pratt

Development Editor: Dan Seiter

Content Project Manager: Jennifer Feltri-George

Product Assistant: Gillian Daniels

Senior Market Development Manager: Eric La Scola

Marketing Manager: Gretchen Swann

Art Director: Cheryl Pearl, GEX Publishing Services

Text Designer: GEX Publishing Services

Cover Designer: GEX Publishing Services

Image Credit: © Kasia/Shutterstock.com

Manufacturing Planner: Julio Esperas

Copyeditor: Michael Beckett

Proofreader: Lisa Weidenfeld

Indexer: Alexandra Nickerson

Compositor: Integra

Library of Congress Control Number: 2013956197

ISBN-13: 978-1-285-77671-2

Cengage Learning
200 First Stamford Place, 4th Floor
Stamford, CT 06902
USA

Cengage Learning is a leading provider of customized learning solutions with office locations around the globe, including Singapore, the United Kingdom, Australia, Mexico, Brazil, and Japan. Locate your local office at: **www.cengage.com/global**

Cengage Learning products are represented in Canada by Nelson Education, Ltd.

Purchase any of our products at your local college store or at our preferred online store: **www.cengagebrain.com**

Printed in the United States of America
1 2 3 4 5 6 7 17 16 15 14 13

CENGAGE brain .com

Buy. Rent. Access.

Access student data files and other study tools on **cengagebrain.com**.

For detailed instructions visit
http://solutions.cengage.com/ctdownloads/

Store your Data Files on a USB drive for maximum efficiency in organizing and working with the files.

Macintosh users should use a program to expand WinZip or PKZip archives. Ask your instructor or lab coordinator for assistance.

Preface

Programming Logic and Design, Comprehensive, Eighth Edition provides the beginning programmer with a guide to developing structured program logic. This textbook assumes no programming language experience. The writing is nontechnical and emphasizes good programming practices. The examples are business examples; they do not assume mathematical background beyond high school business math. Additionally, the examples illustrate one or two major points; they do not contain so many features that students become lost following irrelevant and extraneous details.

The examples in this book have been created to provide students with a sound background in logic, no matter what programming languages they eventually use to write programs. This book can be used in a stand-alone logic course that students take as a prerequisite to a programming course, or as a companion book to an introductory programming text using any programming language.

Organization and Coverage

Programming Logic and Design, Comprehensive, Eighth Edition introduces students to programming concepts and enforces good style and logical thinking. General programming concepts are introduced in Chapter 1. Chapter 2 discusses using data and introduces two important concepts: modularization and creating high-quality programs. It is important to emphasize these topics early so that students start thinking in a modular way and concentrate on making their programs efficient, robust, easy to read, and easy to maintain.

Chapter 3 covers the key concepts of structure, including what structure is, how to recognize it, and most importantly, the advantages to writing structured programs. This chapter's content is unique among programming texts. The early overview of structure presented here gives students a solid foundation in thinking in a structured way.

Chapters 4, 5, and 6 explore the intricacies of decision making, looping, and array manipulation. Chapter 7 provides details of file handling so students can create programs that process a significant amount of data.

In Chapters 8 and 9, students learn more advanced techniques in array manipulation and modularization. Chapters 10 and 11 provide a thorough yet accessible introduction to concepts and terminology used in object-oriented programming. Students learn about classes, objects, instance and static class members, constructors, destructors, inheritance, and the advantages of object-oriented thinking.

Chapter 12 explores additional object-oriented programming issues: event-driven GUI programming, multithreading, and animation. Chapter 13 discusses system design issues and details the features of the Unified Modeling Language. Chapter 14 is a thorough introduction to important database concepts that business programmers should understand.

Four appendices instruct students in working with numbering systems, large unstructured programs, print charts, and post-test loops and case structures.

Programming Logic and Design combines text explanation with flowcharts and pseudocode examples to provide students with alternative means of expressing structured logic. Numerous detailed, full-program exercises at the end of each chapter illustrate the concepts explained within the chapter, and reinforce understanding and retention of the material presented.

Programming Logic and Design distinguishes itself from other programming logic books in the following ways:

- It is written and designed to be non-language specific. The logic used in this book can be applied to any programming language.
- The examples are everyday business examples; no special knowledge of mathematics, accounting, or other disciplines is assumed.
- The concept of structure is covered earlier than in many other texts. Students are exposed to structure naturally, so they will automatically create properly designed programs.
- Text explanation is interspersed with both flowcharts and pseudocode so students can become comfortable with these logic development tools and understand their interrelationship. Screen shots of running programs also are included, providing students with a clear and concrete image of the programs' execution.
- Complex programs are built through the use of complete business examples. Students see how an application is constructed from start to finish instead of studying only segments of programs.

Features

This text focuses on helping students become better programmers and understand the big picture in program development through a variety of key features. In addition to chapter Objectives, Summaries, and Key Terms, these useful features will help students regardless of their learning style.

`not eof?` question is asked. If it is not the end of input data, then the program gets a number, doubles it, and displays it. Then, if the `not eof?` condition remains true, the program gets another number, doubles it, and displays it. The program might continue while many numbers are input. At some point, the input number will represent the `eof` condition; for example, the program might have been written to recognize the value *0* as the program-terminating value. After the `eof` value is entered, its condition is not immediately tested. Instead, a result is calculated and displayed one last time before the loop-controlling question is asked again. If the program was written to recognize `eof` when `originalNumber` is 0, then an extraneous answer of 0 will be displayed before the program ends. Depending on the language you are using and on the type of input being used, the results might be worse: The program might terminate by displaying an error message or the value output might be indecipherable garbage. In any case, this last output is superfluous—no value should be doubled and output after the `eof` condition is encountered.

As a general rule, a program-ending test should always come immediately after an input statement because that's the earliest point at which it can be evaluated. Therefore, the best solution to the number-doubling problem remains the one shown in Figure 3-16—the structured solution containing the priming input statement.

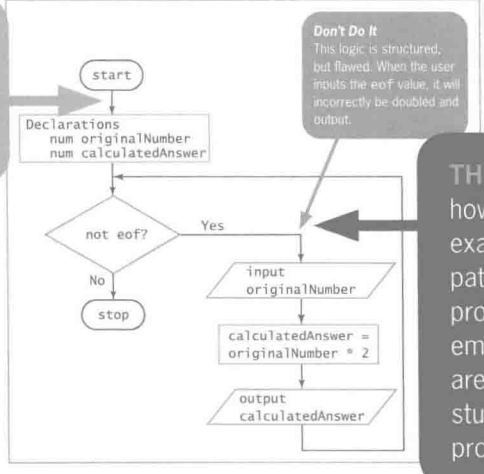


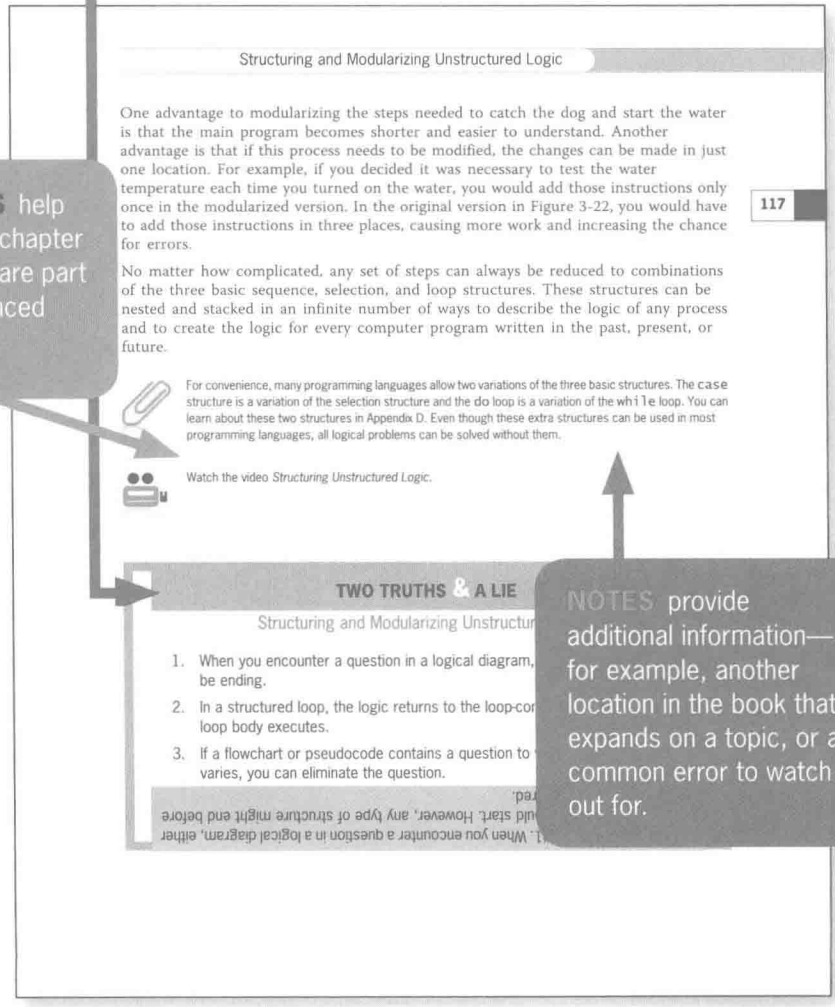
Figure 3-17 Structured but incorrect solution to the number-doubling problem

FLOWCHARTS, figures, and illustrations provide the reader with a visual learning experience.

THE DON'T DO IT ICON illustrates how NOT to do something—for example, having a dead code path in a program. This icon provides a visual jolt to the student, emphasizing that particular figures are NOT to be emulated and making students more careful to recognize problems in existing code.

TWO TRUTHS & A LIE mini quizzes appear after each chapter section, with answers provided. The quiz contains three statements based on the preceding section of text—two statements are true and one is false. Answers give immediate feedback without "giving away" answers to the multiple-choice questions and programming problems later in the chapter. Students also have the option to take these quizzes electronically through the enhanced CourseMate site.

VIDEO LESSONS help explain important chapter concepts. Videos are part of the text's enhanced CourseMate site.



Structuring and Modularizing Unstructured Logic

One advantage to modularizing the steps needed to catch the dog and start the water is that the main program becomes shorter and easier to understand. Another advantage is that if this process needs to be modified, the changes can be made in just one location. For example, if you decided it was necessary to test the water temperature each time you turned on the water, you would add those instructions only once in the modularized version. In the original version in Figure 3-22, you would have to add those instructions in three places, causing more work and increasing the chance for errors.

117

No matter how complicated, any set of steps can always be reduced to combinations of the three basic sequence, selection, and loop structures. These structures can be nested and stacked in an infinite number of ways to describe the logic of any process and to create the logic for every computer program written in the past, present, or future.

For convenience, many programming languages allow two variations of the three basic structures. The `case` structure is a variation of the selection structure and the `do` loop is a variation of the `while` loop. You can learn about these two structures in Appendix D. Even though these extra structures can be used in most programming languages, all logical problems can be solved without them.

Watch the video *Structuring Unstructured Logic*.

TWO TRUTHS & A LIE

Structuring and Modularizing Unstructur

1. When you encounter a question in a logical diagram, be ending.
2. In a structured loop, the logic returns to the loop-cor loop body executes.
3. If a flowchart or pseudocode contains a question to varies, you can eliminate the question.

NOTES provide additional information—for example, another location in the book that expands on a topic, or a common error to watch out for.

Assessment

PROGRAMMING EXERCISES provide opportunities to practice concepts. These exercises increase in difficulty and allow students to explore logical programming concepts. Each exercise can be completed using flowcharts, pseudocode, or both. In addition, instructors can assign the exercises as programming problems to be coded and executed in a particular programming language.

A **loop body** is the set of actions that occur within a loop.

A **while loop** is a structure that continues to repeat a process while some condition remains true.

Repetition and **iteration** are alternate names for a loop structure.

A **while_do loop** is an alternate name for a **while** loop.

Stacking structures is the act of attaching structures end to end.

Nesting structures is the act of placing a structure within another structure.

A **block** is a group of statements that executes as a single unit.

A **priming input** or **priming read** is the statement that reads the first input prior to starting a structured loop that uses the data.

Goto-less programming is a name to describe structur programmers do not use a "go to" statement.

Exercises

Review Questions

1. Snarled program logic is called _____
   a. snake
   b. string
2. The three structures of structured program
   a. sequence, selection, and loop
   b. selection, loop, and iteration
3. A sequence structure can contain _____
   a. only one task
   b. exactly three tasks
4. Which of the following is *not* another term
   a. decision structure
   b. loop structure
5. The structure that tests a condition, takes ac the condition again can be called all of the
   a. iteration
   b. loop

Programming Exercises

1. In Figure 3-10, the process of buying and planting flowers in the spring was shown using the same structures as the generic example in Figure 3-9. Use the same logical structure as in Figure 3-9 to create a flowchart or pseudocode that describes some other process you know.
2. Each of the flowchart segments in Figure 3-24 is unstructured. Redraw each segment so that it does the same thing but is structured.

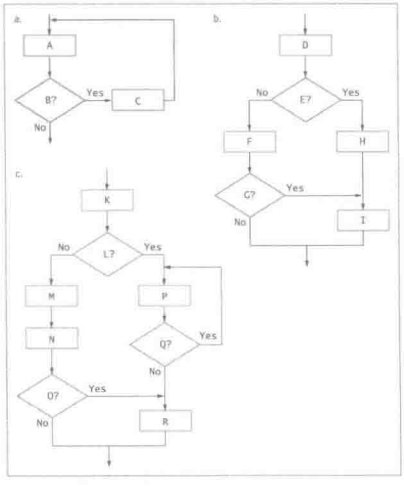


Figure 3-24 Flowcharts for Exercise 2 *(continues)*

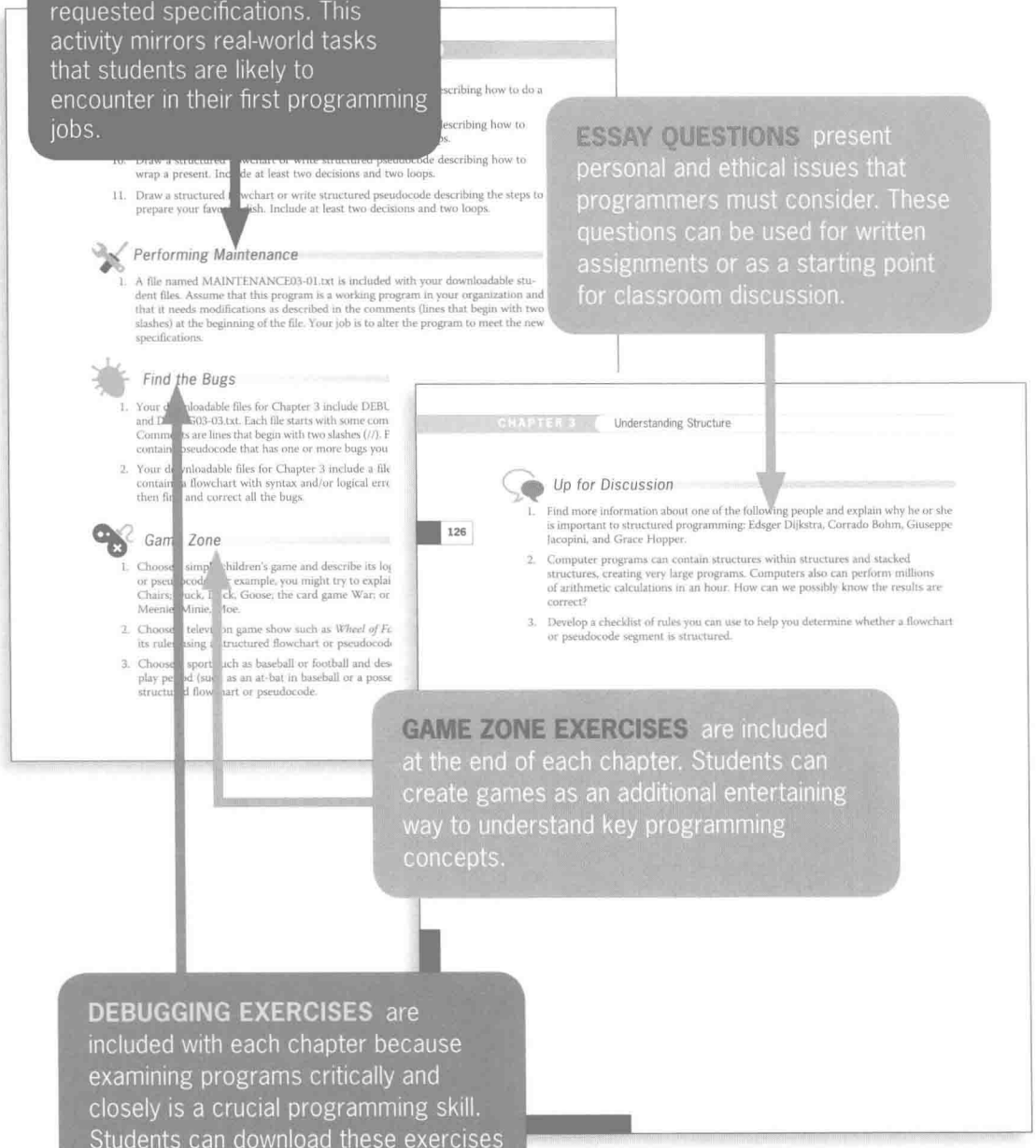
REVIEW QUESTIONS test student comprehension of the major ideas and techniques presented. Twenty questions follow each chapter.

PERFORMING MAINTENANCE exercises ask students to modify working logic based on new requested specifications. This activity mirrors real-world tasks that students are likely to encounter in their first programming jobs.

ESSAY QUESTIONS present personal and ethical issues that programmers must consider. These questions can be used for written assignments or as a starting point for classroom discussion.

GAME ZONE EXERCISES are included at the end of each chapter. Students can create games as an additional entertaining way to understand key programming concepts.

DEBUGGING EXERCISES are included with each chapter because examining programs critically and closely is a crucial programming skill. Students can download these exercises at *www.cengagebrain.com* and through the CourseMate available for this text. These files are also available to instructors at *sso.cengage.com*.



scribing how to do a

escribing how to

os.

wchart or write structured pseudocode describing how to wrap a present. Include at least two decisions and two loops.

11. Draw a structured flowchart or write structured pseudocode describing the steps to prepare your favorite dish. Include at least two decisions and two loops.

Performing Maintenance

1. A file named MAINTENANCE03-01.txt is included with your downloadable student files. Assume that this program is a working program in your organization and that it needs modifications as described in the comments (lines that begin with two slashes) at the beginning of the file. Your job is to alter the program to meet the new specifications.

Find the Bugs

1. Your downloadable files for Chapter 3 include DEBU and DEBUG03-03.txt. Each file starts with some com Comments are lines that begin with two slashes (//). F contain pseudocode that has one or more bugs you
2. Your downloadable files for Chapter 3 include a file contain a flowchart with syntax and/or logical errc then fix and correct all the bugs.

Game Zone

1. Choose a simple children's game and describe its log or pseudocode. For example, you might try to explai Chairs; Duck, Duck, Goose; the card game War; or Meenie Minie Moe.
2. Choose a television game show such as *Wheel of Fo* its rules using a structured flowchart or pseudocod
3. Choose a sport such as baseball or football and des play period (such as an at-bat in baseball or a posse structured flowchart or pseudocode.

Up for Discussion

1. Find more information about one of the following people and explain why he or she is important to structured programming: Edsger Dijkstra, Corrado Bohm, Giuseppe Jacopini, and Grace Hopper.
2. Computer programs can contain structures within structures and stacked structures, creating very large programs. Computers also can perform millions of arithmetic calculations in an hour. How can we possibly know the results are correct?
3. Develop a checklist of rules you can use to help you determine whether a flowchart or pseudocode segment is structured.

Other Features of the Text

This edition of the text includes many features to help students become better programmers and understand the big picture in program development.

- **Clear explanations**. The language and explanations in this book have been refined over eight editions, providing the clearest possible explanations of difficult concepts.
- **Emphasis on structure**. More than its competitors, this book emphasizes structure. Chapter 3 provides an early picture of the major concepts of structured programming.
- **Emphasis on modularity**. From the second chapter, students are encouraged to write code in concise, easily manageable, and reusable modules. Instructors have found that modularization should be encouraged early to instill good habits and a clearer understanding of structure.
- **Objectives**. Each chapter begins with a list of objectives so the student knows the topics that will be presented in the chapter. In addition to providing a quick reference to topics covered, this feature provides a useful study aid.
- **Chapter summaries**. Following each chapter is a summary that recaps the programming concepts and techniques covered in the chapter.
- **Key terms**. Each chapter lists key terms and their definitions; the list appears in the order the terms are encountered in the chapter. A glossary at the end of the book lists all the key terms in alphabetical order, along with working definitions.

CourseMate

The more you study, the better the results. Make the most of your study time by accessing everything you need to succeed in one place. Read your textbook, review flashcards, watch videos, and take practice quizzes online. CourseMate goes beyond the book to deliver what you need! Learn more at *www.cengage.com/coursemate*.

The *Programming Logic and Design* CourseMate includes:

- **Video Lessons**. Designed and narrated by the author, videos in each chapter explain and enrich important concepts.
- **Two Truths & A Lie**, **Debugging Exercises**, and **Performing Maintenance**. Complete popular exercises from the text online.
- **An interactive eBook**. Highlighting and note-taking, flashcards, quizzing, study games, and more.

Instructors may add CourseMate to the textbook package, or students may purchase CourseMate directly at *www.cengagebrain.com*.

Instructor Resources

The following teaching tools are available to the instructor for download through our Instructor Companion Site at *sso.cengage.com*.

- **Electronic Instructor's Manual**. The Instructor's Manual follows the text chapter by chapter to assist in planning and organizing an effective, engaging course. The manual includes learning objectives, chapter overviews, lecture notes, ideas for classroom activities, and abundant additional resources. A sample course syllabus is also available.
- **PowerPoint Presentations**. This text provides PowerPoint slides to accompany each chapter. Slides are included to guide classroom presentation, to make available to students for chapter review, or to print as classroom handouts.
- **Solutions**. Solutions to review questions and exercises are provided to assist with grading.
- **Test Bank®**. Cengage Learning Testing Powered by Cognero is a flexible, online system that allows you to:
  - author, edit, and manage test bank content from multiple Cengage Learning solutions
  - create multiple test versions in an instant
  - deliver tests from your LMS, your classroom, or anywhere you want

Additional Options

- **Visual Logic™ software**. Visual Logic is a simple but powerful tool for teaching programming logic and design without traditional high-level programming language syntax. Visual Logic also interprets and executes flowcharts, providing students with immediate and accurate feedback.
- **PAL (Programs to Accompany) Guides**. Together with *Programming Logic and Design*, these brief books, or PAL Guides, provide an excellent opportunity to learn the fundamentals of programming while gaining exposure to a programming language. PAL guides are available for C++, Java, and Visual Basic; please contact your sales rep for more information on how to add the PAL guides to your course.

Acknowledgments

I would like to thank all of the people who helped to make this book a reality, especially Dan Seiter, Development Editor; Alyssa Pratt, Senior Content Developer; Jim Gish, Senior Product Manager; and Jennifer Feltri-George, Content Project Manager. I am grateful to be able to work with so many fine people who are dedicated to producing quality instructional materials.

I am indebted to the many reviewers who provided helpful and insightful comments during the development of this book, including Gail Gehrig, Florida State College at Jacksonville; Yvonne Leonard, Coastal Carolina Community College; and Meri Winchester, McHenry County College.

Thanks, too, to my husband, Geoff, and our daughters, Andrea and Audrey, for their support. This book, as were all its previous editions, is dedicated to them.

–Joyce Farrell

Contents