Preface

Welcome to *Starting Out with Programming Logic and Design*, Second Edition. This book uses a language-independent approach to teach programming concepts and problem-solving skills, without assuming any previous programming experience. By using easy-to-understand pseudocode, flowcharts, and other tools, the student learns how to design the logic of programs without the complication of language syntax.

Fundamental topics such as data types, variables, input, output, control structures, modules, functions, arrays, and files are covered as well as object-oriented concepts, GUI development, and event-driven programming. As with all the books in the *Starting Out With . . .* series, this text is written with clear, easy-to-understand language that students find friendly and inviting.

Each chapter presents a multitude of program design examples. Short examples that highlight specific programming topics are provided, as well as more involved examples that focus on problem solving. Each chapter includes at least one case study that provides step-by-step analysis of a specific problem and demonstrates a solution to that problem.

This book is ideal for a programming logic course that is taught as a precursor to a language-specific introductory programming course, or for the first part of an introductory programming course in which a specific language is taught.

Changes in the Second Edition

This book's pedagogy, organization, and clear writing style remain the same as in the previous edition. Many improvements have been made, which are summarized here:

- **Online VideoNotes**

  An extensive series of online VideoNotes have been developed to accompany this text. Throughout the book, VideoNote icons alert the student to videos covering specific topics. Additionally, one programming exercise at the end of each chapter now has an accompanying VideoNote explaining how to develop the problem's solution. The videos are available at `www.pearsonhighered.com/gaddis`.

- **Programming Language Companions**

  Programming language companions specifically designed to accompany the Second Edition of this textbook are available for download. The companions introduce the Java™, Python®, and Visual Basic® programming languages, and correspond on a chapter-by-chapter basis with the textbook. Many of the pseudocode programs that appear in the textbook also appear in the companions, implemented in a specific programming language. The programming language companions are available at `www.pearsonhighered.com/gaddis`.

- **New Chapter on Text Processing**

  Chapter 12 in the Second Edition is a new chapter on Text Processing. This chapter discusses techniques for processing strings at the character level. Common library functions for processing characters and text are also discussed.

- **Additional Topics in Chapter 8: Arrays**

  Chapter 8: Arrays has a new section on partially filled arrays, and a new optional section on the `For Each` Loop.

- **Additional Programming Problems**

  Additional programming problems have been added to Chapters 4, 5, and 6. Several of these problems are simple games that will challenge and motivate students.

Brief Overview of Each Chapter

Chapter 1: Introduction to Computers and Programming

This chapter begins by giving a concise and easy-to-understand explanation of how computers work, how data is stored and manipulated, and why we write programs in high-level languages.

Chapter 2: Input, Processing, and Output

This chapter introduces the program development cycle, data types, variables, and sequence structures. The student learns to use pseudocode and flowcharts to design simple programs that read input, perform mathematical operations, and produce screen output.

Chapter 3: Modules

This chapter demonstrates the benefits of modularizing programs and using the top-down design approach. The student learns to define and call modules, pass arguments to modules, and use local variables. Hierarchy charts are introduced as a design tool.

Chapter 4: Decision Structures and Boolean Logic

In this chapter students explore relational operators and Boolean expressions and are shown how to control the flow of a program with decision structures. The `If-Then`, `If-Then-Else`, and `If-Then-Else If` statements are covered. Nested decision structures, logical operators, and the case structure are also discussed.

Chapter 5: Repetition Structures

This chapter shows the student how to use loops to create repetition structures. The `While`, `Do-While`, `Do-Until`, and `For` loops are presented. Counters, accumulators, running totals, and sentinels are also discussed.

Chapter 6: Functions

This chapter begins by discussing common library functions, such as those for generating random numbers. After learning how to call library functions and how to use values returned by functions, the student learns how to define and call his or her own functions.

Chapter 7: Input Validation

This chapter discusses the importance of validating user input. The student learns to write input validation loops that serve as error traps. Defensive programming and the importance of anticipating obvious as well as unobvious errors is discussed.

Chapter 8: Arrays

In this chapter the student learns to create and work with one- and two-dimensional arrays. Many examples of array processing are provided including examples illustrating how to find the sum, average, and highest and lowest values in an array, and how to sum the rows, columns, and all elements of a two-dimensional array. Programming techniques using parallel arrays are also demonstrated.

Chapter 9: Sorting and Searching Arrays

In this chapter the student learns the basics of sorting arrays and searching for data stored in them. The chapter covers the bubble sort, selection sort, insertion sort, and binary search algorithms.

Chapter 10: Files

This chapter introduces sequential file input and output. The student learns to read and write large sets of data, store data as fields and records, and design programs that work with both files and arrays. The chapter concludes by discussing control break processing.

Chapter 11: Menu-Driven Programs

In this chapter the student learns to design programs that display menus and execute tasks according to the user's menu selection. The importance of modularizing a menu-driven program is also discussed.

Chapter 12: Text Processing

This chapter discusses text processing at a detailed level. Algorithms that step through the individual characters in a string are discussed, and several common library functions for character and text processing are introduced.

Chapter 13: Recursion

This chapter discusses recursion and its use in problem solving. A visual trace of recursive calls is provided, and recursive applications are discussed. Recursive algorithms for many tasks are presented, such as finding factorials, finding a greatest common denominator (GCD), summing a range of values in an array, and performing a binary search. The classic Towers of Hanoi example is also presented.

Chapter 14: Object-Oriented Programming

This chapter compares procedural and object-oriented programming practices. It covers the fundamental concepts of classes and objects. Fields, methods, access specification, constructors, accessors, and mutators are discussed. The student learns how to model classes with UML and how to find the classes in a particular problem.

Chapter 15: GUI Applications and Event-Driven Programming

This chapter discusses the basic aspects of designing a GUI application. Building graphical user interfaces with visual design tools (such as Visual Studio® or NetBeans™) is discussed. The student learns how events work in a GUI application and how to write event handlers.

Appendix A: ASCII/Unicode Characters

This appendix lists the ASCII character set, which is the same as the first 127 Unicode character codes.

Appendix B: Flowchart Symbols

This appendix shows the flowchart symbols that are used in this book.

Appendix C: Answers to Checkpoint Questions

This appendix provides answers to the Checkpoint questions that appear throughout the text, and is located on the CD that accompanies this book.[1]

Organization of the Text

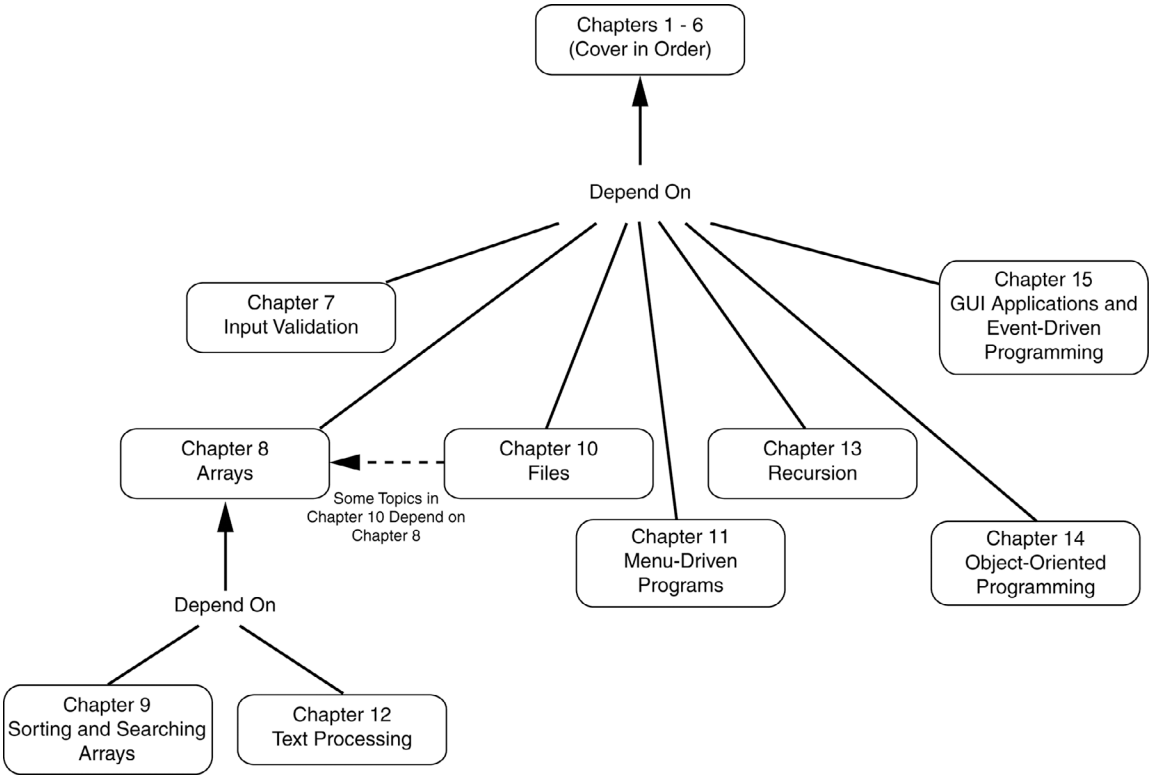
The text teaches programming logic and design in a step-by-step manner. Each chapter covers a major set of topics and builds knowledge as students progress through the book. Although the chapters can be easily taught in their existing sequence, there is some flexibility. Figure P-1 shows chapter dependencies. Each box represents a chapter or a group of chapters. A chapter to which an arrow points must be covered before the chapter from which the arrow originates. The dotted line indicates that only a portion of Chapter 10 depends on information presented in Chapter 8.

Features of the Text

Concept Statements. Each major section of the text starts with a concept statement. This statement concisely summarizes the main point of the section.

[1]If a CD did not come with your book or you can't locate your CD, you can also visit `http://www.aw.com/cssupport/` to access this appendix.

Figure P-1 Chapter Dependencies



Example Programs. Each chapter has an abundant number of complete and partial example programs, each designed to highlight the current topic. Pseudocode, flowcharts, and other design tools are used in the example programs.

Case Studies. Each chapter has one or more *In the Spotlight* case studies that provide detailed, step-by-step analysis of problems, and show the student how to solve them.

VideoNote

VideoNotes. A series of online videos, developed specifically for this book, are available for viewing at `www.pearsonhighered.com/gaddis`. Icons appear throughout the text alerting the student to videos about specific topics.

NOTE: Notes appear at several places throughout the text. They are short explanations of interesting or often misunderstood points relevant to the topic at hand.

TIP: Tips advise the student on the best techniques for approaching different programming or animation problems.

WARNING! Warnings caution students about programming techniques or practices that can lead to malfunctioning programs or lost data.

Programming Language Companions. Many of the pseudocode programs shown in this book have also been written in Java, Python, and Visual Basic. These programs appear in the programming language companions that are available at `www.pearsonhighered.com/gaddis`. Icons appear next to each pseudocode program that also appears in the language companions.

Checkpoints. Checkpoints are questions placed at intervals throughout each chapter. They are designed to query the student's knowledge quickly after learning a new topic.

Review Questions. Each chapter presents a thorough and diverse set of Review Questions and exercises. They include Multiple Choice, True/False, Short Answer, and Algorithm Workbench.

Programming Exercises. Each chapter offers a pool of Programming Exercises designed to solidify the student's knowledge of the topics currently being studied.

Supplements

Student Resource CD

This CD includes:

- Answers to Checkpoint Questions (Appendix C)
- RAPTOR, a flowchart-based programming environment

If a CD did not come with your book or you can't locate your CD, you can visit `http://www.aw.com/cssupport/` to access these supplements.

Programming Language Companions

Programming language companions specifically designed to accompany the Second Edition of this textbook are available for download. The companions introduce the Java, Python, and Visual Basic programming languages, and correspond on a chapter-by-chapter basis with the textbook. Many of the pseudocode programs that appear in the textbook also appear in the companions, implemented in a specific programming language. The programming language companions are available at `www.pearsonhighered.com/gaddis`.

Instructor Resources

The following supplements are available to qualified instructors only:

- Answers to all of the Review Questions
- Solutions for the Programming Exercises
- PowerPoint® presentation slides for each chapter
- Test bank

Visit the Addison-Wesley Instructor Resource Center (`http://www.pearsonhighered.com/irc`) or send an email to `computing@aw.com` for information on how to access them.

Acknowledgments

There have been many helping hands in the development and publication of this text. I would like to thank the following faculty reviewers:

Reni Abraham
Houston Community College

John P. Buerck
Saint Louis University

Jill Canine
Ivy Tech Community College of Indiana

Steven D. Carver
Ivy Tech Community College of Indiana

Katie Danko
Grand Rapids Community College

Coronicca Oliver
Coastal Georgia Community College

Dale T. Pickett
Baker College of Clinton Township

Tonya Pierce
Ivy Tech Community College

Larry Strain
Ivy Tech Community College–Bloomington

Donald Stroup
Ivy Tech Community College

Jim Turney
Austin Community College

I also want to thank everyone at Pearson Addison-Wesley for making the *Starting Out With . . .* series so successful. I have worked so closely with the team at Pearson Addison-Wesley that I consider them among my closest friends. I am extremely grateful that Michael Hirsch is my editor. He and Stephanie Sellinger, editorial assistant, have guided me through the process of revising this book. I am also thankful to have Erin Davis as marketing manager. Her energy and creativity are truly inspiring. The production team worked tirelessly to make this book a reality, and includes Jeff Holcomb, Katelyn Boller, Carol Melville, Linda Knowles, and Dennis Free. Thanks to you all!

Last, but not least, I want to thank my family for all the patience, love, and support they have shown me throughout this and my many other projects.

About the Author

Tony Gaddis is the principal author of the *Starting Out With . . .* series of textbooks. Tony has twenty years of experience teaching computer science courses, primarily at Haywood Community College. He is a highly acclaimed instructor who was previously selected as the North Carolina Community College "Teacher of the Year" and has received the Teaching Excellence award from the National Institute for Staff and Organizational Development. The *Starting Out With . . .* series includes introductory books covering Programming Logic and Design, C++, Java, Microsoft® Visual Basic, C#®, Python, and Alice, all published by Pearson Addison-Wesley.