Welcome to Starting Out with Programming Logic and Design, Third 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 in 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 In the Spotlight section 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 Third 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:

• Detailed guidance for students designing their first program

A new section titled Designing Your First Program has been added to Chapter 2. This section takes the student through the process of analyzing a problem and determining its requirements. The student sees an example of how a program’s input, processing, and output can be determined, as a prelude to writing pseudocode and drawing flowcharts.

Also, a new In the Spotlight section has been added to Chapter 2 to show the student how to examine the steps that are taken to manually perform a calculation (determining cell phone overage fees), and then convert those steps to a computer algorithm.

• New Debugging Exercises

A new set of Debugging Exercises have been added to most of the chapters. The student examines a set of pseudocode algorithms and identifies logical errors.
Preface

• **Greater consistency between flowcharts and pseudocode**
  Throughout the book, many of the flowcharts have been revised so they appear more consistent with the pseudocode.

• **Expanded coverage of nested repetition structures**
  In Chapter 5 the section on nested loops has been expanded with an additional example.

• **Additional VideoNotes for repetition structures**
  New VideoNotes have been added for the Do-While and For loops in Chapter 5.

• **File specification documentation and print spacing charts**
  File specification documentation and print spacing charts are now discussed in Chapter 10.

• **New pseudocode quick reference guide**
  A quick reference guide to the pseudocode used in the book has been added as Appendix C.

• **New Programming Language Companions**
  New language companions have been added for Python 3 and C++. All of the book’s language companions are available on the book’s resource site at www.pearsonhighered.com/gaddis.

**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: Pseudocode Reference
This appendix provides a quick reference for the pseudocode language that is used in the book.

Appendix D: Answers to Checkpoint Questions
This appendix provides answers to the Checkpoint questions that appear throughout the text, and can be downloaded from the CD that accompanies this book or from the book’s online resource page at www.pearsonhighered.com/gaddis.

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.

**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.

**In the Spotlight.** 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.

**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.
Preface

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.

Debugging Exercises. Most chapters provide a set of debugging exercises in which the student examines a set of pseudocode algorithms and identifies logical errors.

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 Online Resources

Many student resources are available for this book from the publisher. The following items are available on the Gaddis Series resource page at www.pearsonhighered.com/gaddis:

- Access to the book’s companion 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 has an accompanying VideoNote explaining how to develop the problem’s solution.

- Access to the Language Companions for Python, Java, Visual Basic, and C++

  Programming language companions specifically designed to accompany the Third Edition of this textbook are available for download. The companions introduce the Java™, Python®, Visual Basic®, and C++ 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.

• **A link to download the RAPTOR flowcharting environment**

  RAPTOR is a flowchart-based programming environment developed by the US Air Force Academy Department of Computer Science.

**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 Pearson Instructor Resource Center (http://www.pearsonhighered.com/irc) or send an email to computing@aw.com for information on how to access them.