

starting out with >>> **PROGRAMMING
LOGIC AND DESIGN**

SIXTH EDITION



TONY GADDIS



Sixth
Edition

Starting Out with

Programming Logic & Design

This page intentionally left blank

Sixth
Edition

Starting Out with

Programming Logic & Design

Tony Gaddis

Haywood Community College



Content Development: Dawn Murrin
Content Management: Tracy Johnson
Content Production: Ishan Chaudhary and Carole Snyder
Product Management: Holly Stark
Product Marketing: Wayne Stevens
Rights and Permissions: Anjali Singh

Please contact <https://support.pearson.com/getsupport/s/> with any queries on this content.

Cover Image by Patrick de grijs/123RF.

Microsoft and/or its respective suppliers make no representations about the suitability of the information contained in the documents and related graphics published as part of the services for any purpose. All such documents and related graphics are provided “as is” without warranty of any kind. Microsoft and/or its respective suppliers hereby disclaim all warranties and conditions with regard to this information, including all warranties and conditions of merchantability, whether express, implied or statutory, fitness for a particular purpose, title and non-infringement. In no event shall Microsoft and/or its respective suppliers be liable for any special, indirect or consequential damages or any damages whatsoever resulting from loss of use, data or profits, whether in an action of contract, negligence or other tortious action, arising out of or in connection with the use or performance of information available from the services.

The documents and related graphics contained herein could include technical inaccuracies or typographical errors. Changes are periodically added to the information herein. Microsoft and/or its respective suppliers may make improvements and/or changes in the product(s) and/or the program(s) described herein at any time. Partial screen shots may be viewed in full within the software version specified.

Microsoft® and Windows® are registered trademarks of the Microsoft Corporation in the U.S.A. and other countries. This book is not sponsored or endorsed by or affiliated with the Microsoft Corporation.

Copyright © 2023, 2019 by Pearson Education, Inc. or its affiliates, 221 River Street, Hoboken, NJ 07030. All Rights Reserved. Manufactured in the United States of America. This publication is protected by copyright, and permission should be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise. For information regarding permissions, request forms, and the appropriate contacts within the Pearson Education Global Rights and Permissions department, please visit www.pearsoned.com/permissions/.

Acknowledgments of third-party content appear on the appropriate page within the text.

PEARSON is exclusive trademarks owned by Pearson Education, Inc. or its affiliates in the U.S. and/or other countries.

Unless otherwise indicated herein, any third-party trademarks, logos, or icons that may appear in this work are the property of their respective owners, and any references to third-party trademarks, logos, icons, or other trade dress are for demonstrative or descriptive purposes only. Such references are not intended to imply any sponsorship, endorsement, authorization, or promotion of Pearson’s products by the owners of such marks, or any relationship between the owner and Pearson Education, Inc., or its affiliates, authors, licensees, or distributors.

Library of Congress Cataloging-in-Publication Data

Starting Out with Programming Logic & Design

Library of Congress Cataloging in Publication Control Number: 2021057225

ScoutAutomatedPrintCode



Rental:

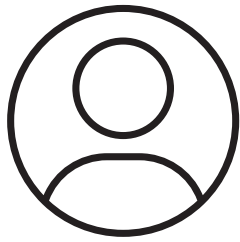
ISBN-10: 0-13-760214-6

ISBN-13: 978-0-13-760214-8

Print offer:

ISBN-10: 0-13-760191-3

ISBN-13: 978-0-13-760191-2



Pearson's Commitment to Diversity, Equity, and Inclusion

Pearson is dedicated to creating bias-free content that reflects the diversity, depth, and breadth of all learners' lived experiences.

We embrace the many dimensions of diversity, including but not limited to race, ethnicity, gender, sex, sexual orientation, socioeconomic status, ability, age, and religious or political beliefs.

Education is a powerful force for equity and change in our world. It has the potential to deliver opportunities that improve lives and enable economic mobility. As we work with authors to create content for every product and service, we acknowledge our responsibility to demonstrate inclusivity and incorporate diverse scholarship so that everyone can achieve their potential through learning. As the world's leading learning company, we have a duty to help drive change and live up to our purpose to help more people create a better life for themselves and to create a better world.

Our ambition is to purposefully contribute to a world where:

- Everyone has an equitable and lifelong opportunity to succeed through learning.
- Our educational content accurately reflects the histories and lived experiences of the learners we serve.
- Our educational products and services are inclusive and represent the rich diversity of learners.
- Our educational content prompts deeper discussions with students and motivates them to expand their own learning (and worldview).

Accessibility

We are also committed to providing products that are fully accessible to all learners. As per Pearson's guidelines for accessible educational Web media, we test and retest the capabilities of our products against the highest standards for every release, following the WCAG guidelines in developing new products for copyright year 2022 and beyond.



You can learn more about Pearson's commitment to accessibility at <https://www.pearson.com/us/accessibility.html>

Contact Us

While we work hard to present unbiased, fully accessible content, we want to hear from you about any concerns or needs with this Pearson product so that we can investigate and address them.



Please contact us with concerns about any potential bias at <https://www.pearson.com/report-bias.html>



For accessibility-related issues, such as using assistive technology with Pearson products, alternative text requests, or accessibility documentation, email the Pearson Disability Support team at disability.support@pearson.com



Brief Contents

	Preface	xiii
	Acknowledgments	xxi
	About the Author	xxv
Chapter 1	Introduction to Computers and Programming	1
Chapter 2	Input, Processing, and Output	27
Chapter 3	Decision Structures and Boolean Logic	103
Chapter 4	Repetition Structures	161
Chapter 5	Modules	227
Chapter 6	Functions	285
Chapter 7	Input Validation	335
Chapter 8	Arrays	353
Chapter 9	Sorting and Searching Arrays	419
Chapter 10	Files	469
Chapter 11	Menu-Driven Programs	543
Chapter 12	Text Processing	595
Chapter 13	Recursion	623
Chapter 14	Object-Oriented Programming	649
Chapter 15	GUI Applications and Event-Driven Programming	715
Appendix A	ASCII/Unicode Characters	747
Appendix B	Flowchart Symbols	749
Appendix C	Pseudocode Reference	751
Appendix D	Converting Decimal Numbers to Binary	765
Appendix E	Answers to Checkpoint Questions	767
	Index	783

Contents

Preface	xiii
Acknowledgments	xxi
About the Author	xxv

Chapter 1 **Introduction to Computers and Programming 1**

1.1	Introduction	1
1.2	Hardware	2
1.3	How Computers Store Data	7
1.4	How a Program Works	12
1.5	Types of Software	20
	Review Questions	21

Chapter 2 **Input, Processing, and Output 27**

2.1	Designing a Program	27
2.2	Output, Input, and Variables	34
2.3	Variable Assignment and Calculations	43
	IN THE SPOTLIGHT: Calculating Cell Phone Overage Fees	47
	IN THE SPOTLIGHT: Calculating a Percentage	49
	IN THE SPOTLIGHT: Calculating an Average	52
	IN THE SPOTLIGHT: Converting a Math Formula to a Programming Statement	55
2.4	Variable Declarations and Data Types	57
2.5	Named Constants	62
2.6	Hand Tracing a Program	64
2.7	Documenting a Program	65
	IN THE SPOTLIGHT: Using Named Constants, Style Conventions, and Comments	66
2.8	Designing Your First Program	68
2.9	Focus on Languages: Java, Python, and C++	72
	Review Questions	92
	Debugging Exercises	97
	Programming Exercises	98

Chapter 3 **Decision Structures and Boolean Logic 103**

3.1 Introduction to Decision Structures	103
IN THE SPOTLIGHT: Using the If-Then Statement	110
3.2 Dual Alternative Decision Structures	113
IN THE SPOTLIGHT: Using the If-Then-Else Statement	114
3.3 Comparing Strings	118
3.4 Nested Decision Structures	122
IN THE SPOTLIGHT: Multiple Nested Decision Structures	125
3.5 The Case Structure	129
IN THE SPOTLIGHT: Using a Case Structure	132
3.6 Logical Operators	134
3.7 Boolean Variables	141
3.8 Focus on Languages: Java, Python, and C++	142
Review Questions	154
Debugging Exercises	158
Programming Exercises	158

Chapter 4 **Repetition Structures 161**

4.1 Introduction to Repetition Structures	161
4.2 Condition-Controlled Loops: While, Do-While, and Do-Until	162
IN THE SPOTLIGHT: Designing a While Loop	167
IN THE SPOTLIGHT: Designing a Do-While Loop	174
4.3 Count-Controlled Loops and the For Statement	179
IN THE SPOTLIGHT: Designing a Count-Controlled Loop with the For Statement	189
4.4 Calculating a Running Total	199
4.5 Sentinels	203
IN THE SPOTLIGHT: Using a Sentinel	204
4.6 Nested Loops	207
4.7 Focus on Languages: Java, Python, and C++	210
Review Questions	219
Debugging Exercises	222
Programming Exercises	223

Chapter 5 **Modules 227**

5.1 Introduction to Modules	227
5.2 Defining and Calling a Module	230
IN THE SPOTLIGHT: Defining and Calling Modules	239
5.3 Local Variables	244
5.4 Passing Arguments to Modules	247
IN THE SPOTLIGHT: Passing an Argument to a Module	251

IN THE SPOTLIGHT: Passing an Argument by Reference	257
5.5 Global Variables and Global Constants	260
IN THE SPOTLIGHT: Using Global Constants	261
5.6 Focus on Languages: Java, Python, and C++	265
Review Questions	276
Debugging Exercises	280
Programming Exercises	281

Chapter 6 **Functions 285**

6.1 Introduction to Functions: Generating Random Numbers	285
IN THE SPOTLIGHT: Using Random Numbers	289
IN THE SPOTLIGHT: Using Random Numbers to Represent Other Values	292
6.2 Writing Your Own Functions	294
IN THE SPOTLIGHT: Modularizing with Functions	300
6.3 More Library Functions	309
6.4 Focus on Languages: Java, Python, and C++	319
Review Questions	326
Debugging Exercises	329
Programming Exercises	330

Chapter 7 **Input Validation 335**

7.1 Garbage In, Garbage Out	335
7.2 The Input Validation Loop	336
IN THE SPOTLIGHT: Designing an Input Validation Loop	338
7.3 Defensive Programming	343
7.4 Focus on Languages: Java, Python, and C++	344
Review Questions	348
Debugging Exercises	350
Programming Exercises	351

Chapter 8 **Arrays 353**

8.1 Array Basics	353
IN THE SPOTLIGHT: Using Array Elements in a Math Expression	360
8.2 Sequentially Searching an Array	367
8.3 Processing the Contents of an Array	373
IN THE SPOTLIGHT: Processing an Array	380
8.4 Parallel Arrays	387
IN THE SPOTLIGHT: Using Parallel Arrays	388
8.5 Two-Dimensional Arrays	392
IN THE SPOTLIGHT: Using a Two-Dimensional Array	395
8.6 Arrays of Three or More Dimensions	399

8.7 Focus on Languages: Java, Python, and C++ 401
Review Questions 411
Debugging Exercises 414
Programming Exercises 415

Chapter 9 Sorting and Searching Arrays 419

9.1 The Bubble Sort Algorithm 419
IN THE SPOTLIGHT: Using the Bubble Sort Algorithm 426
9.2 The Selection Sort Algorithm. 433
9.3 The Insertion Sort Algorithm 439
9.4 The Binary Search Algorithm 445
IN THE SPOTLIGHT: Using the Binary Search Algorithm. 449
9.5 Focus on Languages: Java, Python, and C++ 451
Review Questions 464
Debugging Exercise. 467
Programming Exercises 467

Chapter 10 Files 469

10.1 Introduction to File Input and Output 469
10.2 Using Loops to Process Files. 481
IN THE SPOTLIGHT: Working with Files. 486
10.3 Using Files and Arrays 490
10.4 Processing Records. 491
IN THE SPOTLIGHT: Adding and Displaying Records. 496
IN THE SPOTLIGHT: Searching for a Record. 500
IN THE SPOTLIGHT: Modifying Records. 502
IN THE SPOTLIGHT: Deleting Records 506
10.5 Control Break Logic 509
IN THE SPOTLIGHT: Using Control Break Logic. 510
10.6 Focus on Languages: Java, Python, and C++ 516
Review Questions 536
Debugging Exercises 539
Programming Exercises 540

Chapter 11 Menu-Driven Programs 543

11.1 Introduction to Menu-Driven Programs 543
11.2 Modularizing a Menu-Driven Program 554
11.3 Using a Loop to Repeat the Menu 559
IN THE SPOTLIGHT: Designing a Menu-Driven Program 564
11.4 Multiple-Level Menus 578
11.5 Focus on Languages: Java, Python, and C++ 584

Review Questions	590
Programming Exercises	592
Chapter 12 Text Processing 595	
12.1 Introduction	595
12.2 Character-by-Character Text Processing	597
IN THE SPOTLIGHT: Validating a Password	600
IN THE SPOTLIGHT: Formatting and Unformatting Telephone Numbers	606
12.3 Focus on Languages: Java, Python, and C++	611
Review Questions	617
Debugging Exercises	619
Programming Exercises	620
Chapter 13 Recursion 623	
13.1 Introduction to Recursion	623
13.2 Problem Solving with Recursion	626
13.3 Examples of Recursive Algorithms	630
13.4 Focus on Languages: Java, Python, and C++	640
Review Questions	645
Programming Exercises	647
Chapter 14 Object-Oriented Programming 649	
14.1 Procedural and Object-Oriented Programming	649
14.2 Classes	653
14.3 Using the Unified Modeling Language to Design Classes	664
14.4 Finding the Classes and Their Responsibilities in a Problem	667
IN THE SPOTLIGHT: Finding the Classes in a Problem	667
IN THE SPOTLIGHT: Determining Class Responsibilities	671
14.5 Inheritance	677
14.6 Polymorphism	685
14.7 Focus on Languages: Java, Python, and C++	689
Review Questions	707
Programming Exercises	710
Chapter 15 GUI Applications and Event-Driven Programming 715	
15.1 Graphical User Interfaces	715
15.2 Designing the User Interface for a GUI Program	718
IN THE SPOTLIGHT: Designing a Window	723
15.3 Writing Event Handlers	725

IN THE SPOTLIGHT: Designing an Event Handler. 728
15.4 Designing Apps for Mobile Devices. 731
15.5 Focus on Languages: Java, Python, and C++. 740
Review Questions. 741
Programming Exercises. 744

Appendix A **ASCII/Unicode Characters 747**
Appendix B **Flowchart Symbols 749**
Appendix C **Pseudocode Reference 751**
Appendix D **Converting Decimal Numbers to Binary 765**
Appendix E **Answers to Checkpoint Questions 767**

Index 783

Preface

Welcome to *Starting Out with Programming Logic and Design*, Sixth 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 Sixth Edition

Previous editions of this book introduced modules, which are procedures that do not return a value, in Chapter 3. Feedback from adopters and reviewers indicate that students sometimes have trouble learning about modules before they have been exposed to control structures, such as If statements and loops. In this edition, the chapter on modules has been moved to Chapter 5. Now, the students will learn about control structures, then modules, and then value-returning functions. This improved pedagogy gradually introduces the students to the different ways a program's flow of execution can be directed.

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: 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 4: 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 5: 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 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: Converting Decimal Numbers to Binary

This appendix uses a simple tutorial to demonstrate how to convert a decimal number to binary.

Appendix E: Answers to Checkpoint Questions

This appendix provides answers to the Checkpoint questions that appear throughout the text.

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.

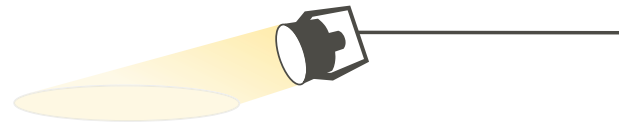
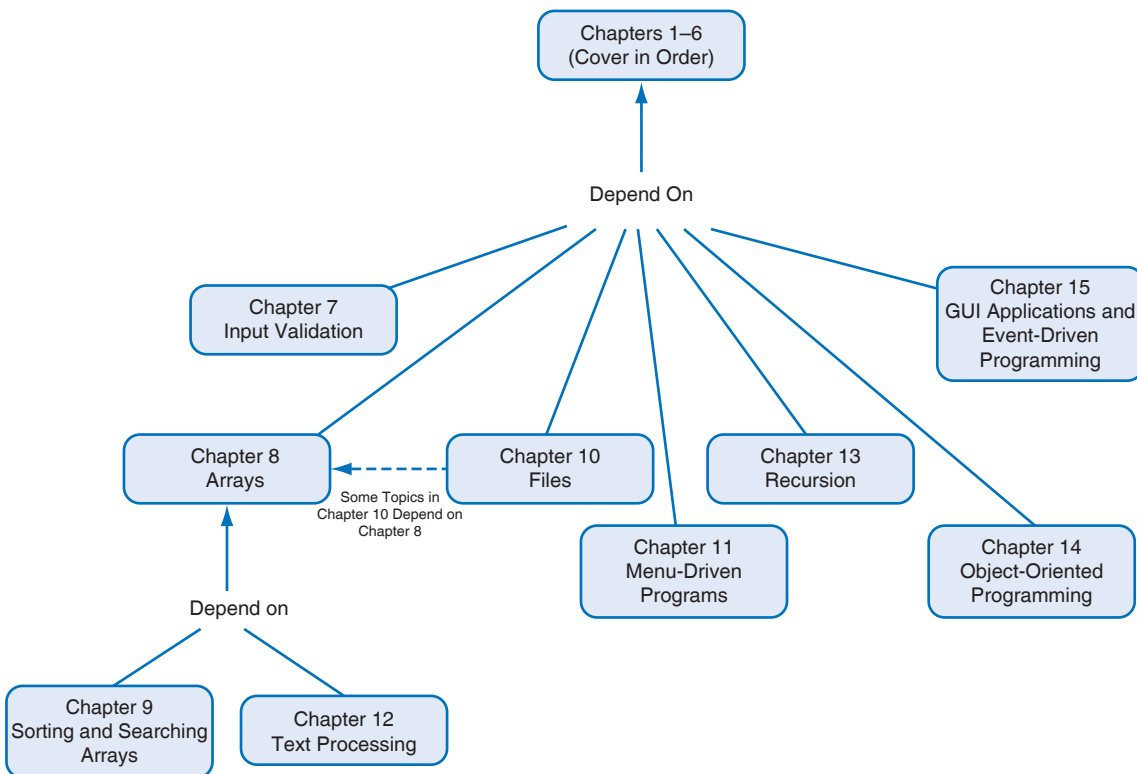


Figure P-1 Chapter dependencies

VideoNotes. A series of online videos, developed specifically for this book, are available for viewing at www.pearson.com/cs-resources. 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 C++. These programs appear in the programming language companions that are available at www.pearson.com/cs-resources. 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.pearson.com/cs-resources:

- **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, and C++**

Programming language companions specifically designed to accompany this textbook are available for download. The companions introduce the Java™, Python®, 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 Flowgorithm flowcharting application**

Flowgorithm is a free application, developed by Devin Cook at Sacramento State University, which allows you to create programs using simple flowcharts. It supports the flowcharting conventions used in this textbook, as well as several other standard conventions. When you create a flowchart with Flowgorithm, you can execute the program and generate Gaddis Pseudocode. You can also generate source code in Java, Python, Visual Basic, C#, Ruby, JavaScript, and several other languages. For more information, see www.flowgorithm.org.

- **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. For more information, see <https://raptor.martincarlisle.com>.

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 www.pearson.com or contact your local Pearson representative for information on how to access them.

This page intentionally left blank

Acknowledgments

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

Reviewers for This Edition

Taz Daughtrey
Central Virginia Community College

Donna Sandsmark
University of California, San Diego

Holly Tajlil
Sacramento State University

Deborah Wilson
Ashland University

Reviewers of Previous Editions

Reni Abraham
Houston Community College

Alan Anderson
Gwinnett Technical College

Cherie Aukland
Thomas Nelson Community College

Steve Browning
Freed Hardeman University

John P. Buerck
Saint Louis University

Jill Canine
Ivy Tech Community College of Indiana

Tony Cantrell
Georgia Northwestern Technical College

Steven D. Carver
Ivy Tech Community College

Stephen Robert Cheskiewicz
Keystone College and Wilkes University

Katie Danko
Grand Rapids Community College

Richard J. Davison
College of the Albemarle

Sameer Dutta
Grambling State University

Norman P. Hahn
Thomas Nelson Community College

John Haley
Athens Technical College

Keith Hallmark
Calhoun Community College

Ronald J. Harkins
Miami University, OH

Dianne Hill
Jackson College

Vai Kumar
Pensacola State College

Coronica Oliver
Coastal Georgia Community College

Robert S. Overall, III
Nashville State Community College

Dale T. Pickett
Baker College of Clinton Township

Tonya Pierce
Ivy Tech Community College

J. Shawn Pope
Tulsa Community College

Maryam Rahnemoonfar
Texas A&M University

Linda Reeser
Arizona Western College

Homayoun Sharafi
Prince George's Community College

Emily Shepard
Central Carolina Community College

Larry Strain
Ivy Tech Community College–Bloomington

Donald Stroup
Ivy Tech Community College

John Thacher
Gwinnett Technical College

Jim Turney
Austin Community College

Scott Vanselow
Edison State College

I would like to thank the faculty, staff, and administration at Haywood Community College for the opportunity to build a career teaching the subjects that I love. I would also like to thank my family and friends for their support in all my projects.

It is a great honor to be published by Pearson, and I am extremely fortunate to have Tracy Johnson as my Content Manager. She and her colleagues Holly Stark, Erin Sullivan, Sandra Rodriguez, Wayne Stevens, Scott Disanno, Bob Engelhardt, Ishan Chaudhary, Carole Snyder, and Mahalakshmi Usha have worked tirelessly to produce and promote this book. Thanks to you all!

This page intentionally left blank

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 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, App Inventor, and Alice, all published by Pearson.

This page intentionally left blank

Sixth
Edition

Starting Out with

Programming Logic & Design

This page intentionally left blank

Introduction to Computers and Programming

TOPICS

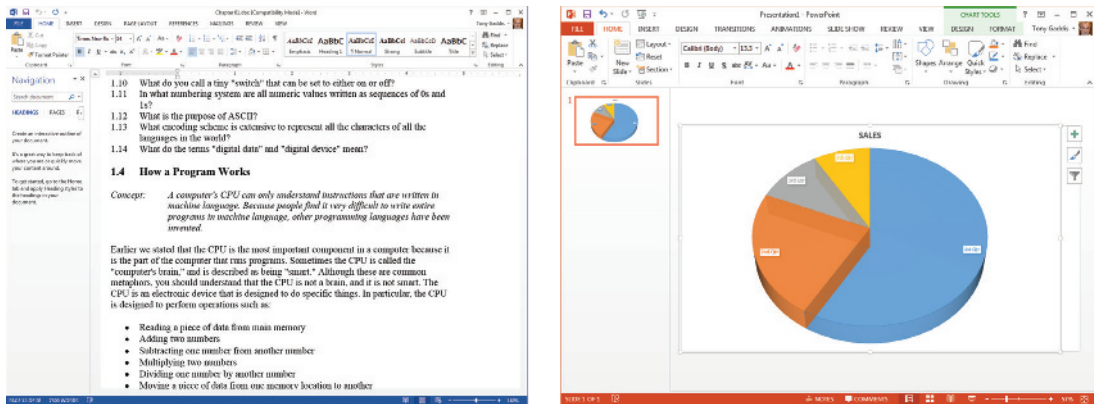
- | | |
|------------------------------|-------------------------|
| 1.1 Introduction | 1.4 How a Program Works |
| 1.2 Hardware | 1.5 Types of Software |
| 1.3 How Computers Store Data | |

1.1

Introduction

Think about some of the different ways that people use computers. In school, students use computers for tasks such as writing papers, searching for articles, sending email, and participating in online classes. At work, people use computers to analyze data, make presentations, conduct business transactions, communicate with customers and coworkers, control machines in manufacturing facilities, and many other things. At home, people use computers for tasks such as paying bills, shopping online, communicating with friends and family, and playing computer games. And don't forget that smart phones, tablets, home automation devices, car navigation systems, and many other devices are computers too. The uses of computers are almost limitless in our everyday lives.

Computers can do such a wide variety of things because they can be programmed. This means that computers are not designed to do just one job, but to do any job that their programs tell them to do. A *program* is a set of instructions that a computer follows to perform a task. For example, Figure 1-1 shows screens from two commonly used programs: Microsoft Word and PowerPoint.

Figure 1-1 Commonly used programs (courtesy of Microsoft Corporation)

Programs are commonly referred to as *software*. Software is essential to a computer because without software, a computer can do nothing. All of the software that we use to make our computers useful is created by individuals known as programmers or software developers. A *programmer*, or *software developer*, is a person with the training and skills necessary to design, create, and test computer programs. Computer programming is an exciting and rewarding career. Today, you will find programmers working in business, medicine, government, law enforcement, agriculture, academics, entertainment, and almost every other field.

This book introduces you to the fundamental concepts of computer programming. Before we begin exploring those concepts, you need to understand a few basic things about computers and how they work. This chapter will build a solid foundation of knowledge that you will continually rely on as you study computer science. First, we will discuss the physical components that computers are commonly made of. Next, we will look at how computers store data and execute programs. Finally, we will discuss the major types of software that computers use.

1.2 Hardware

CONCEPT: The physical devices that a computer is made of are referred to as the computer's hardware. Most computer systems are made of similar hardware devices.

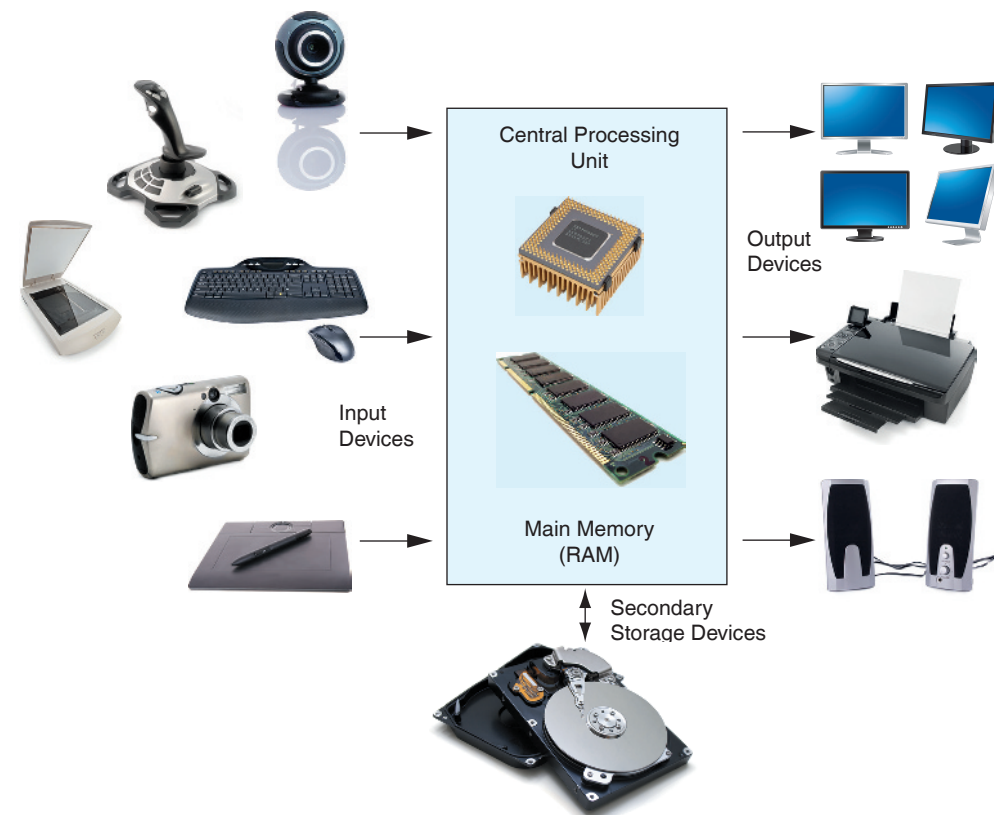
The term *hardware* refers to all of the physical devices, or *components*, that a computer is made of. A computer is not one single device, but a system of devices that all work together. Like the different instruments in a symphony orchestra, each device in a computer plays its own part.

If you have ever shopped for a computer, you've probably seen sales literature listing components such as microprocessors, memory, disk drives, video displays, graphics cards, and so on. Unless you already know a lot about computers, or at least have a friend who

does, understanding what these different components do can be confusing. As shown in Figure 1-2, a typical computer system consists of the following major components:

- The central processing unit (CPU)
- Main memory
- Secondary storage devices
- Input devices
- Output devices

Figure 1-2 Typical components of a computer system



(**Photo credits:** Webcam Iko/Shutterstock, Joystick Nikita Rogul/Shutterstock, Scanner Feng Yu/Shutterstock, Keyboard Chiyacat/Shutterstock, Camera Elkostas/Shutterstock, Tablet Tkemot/Shutterstock, Hard disk Vitaly Korovin/Shutterstock, Speakers StockPhotosArt/Shutterstock, Printer Jovic/Shutterstock, Monitors Art gallery/Shutterstock, RAM Peter Guess/Shutterstock, Chip Aquila/Shutterstock).

Let's take a closer look at each of these components.

The CPU

When a computer is performing the tasks that a program tells it to do, we say that the computer is *running* or *executing* the program. The *central processing unit*, or *CPU*, is the part of a computer that actually runs programs. (The CPU is often referred to as the *processor*.) The CPU is the most important component in a computer because without it, the computer could not run software.

In the earliest computers, CPUs were huge devices made of electrical and mechanical components such as vacuum tubes and switches. Figure 1-3 shows such a device. The two women in the photo are working with the historic ENIAC computer. The *ENIAC*, considered by many to be the world's first programmable electronic computer, was built in 1945 to calculate artillery ballistic tables for the U.S. Army. This machine, which was primarily one big CPU, was 8 feet tall, 100 feet long, and weighed 30 tons.

Today, CPUs are small chips known as *microprocessors*. Figure 1-4 shows a photo of a lab technician holding a modern-day microprocessor. In addition to being much

Figure 1-3 The ENIAC computer (courtesy of US Army Center of Military History)

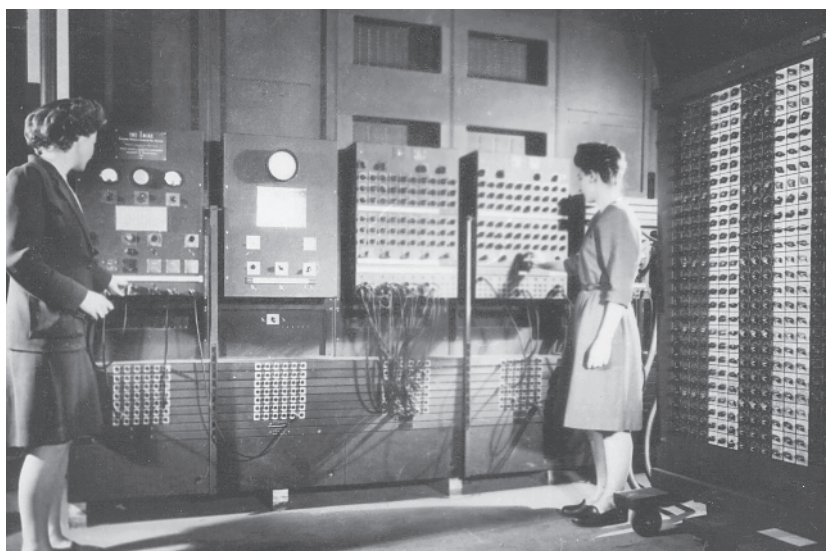


Figure 1-4 A lab technician holds a modern microprocessor (courtesy of Chris Ryan/OJO Images/Getty Images)



smaller than the old electro-mechanical CPUs in early computers, microprocessors are also much more powerful.

Main Memory

You can think of *main memory* as the computer's work area. This is where the computer stores a program while the program is running, as well as the data that the program is working with. For example, suppose you are using a word processing program to write an essay for one of your classes. While you do this, both the word processing program and the essay are stored in main memory.

Main memory is commonly known as *random-access memory*, or *RAM*. It is called this because the CPU is able to quickly access data stored at any random location in RAM. RAM is usually a *volatile* type of memory that is used only for temporary storage while a program is running. When the computer is turned off, the contents of RAM are erased. Inside your computer, RAM is stored in chips, similar to the ones shown in Figure 1-5.

Figure 1-5 Memory chips (photo © Garsya/Shutterstock)



NOTE: Another type of memory that is stored in chips inside the computer is *read-only memory*, or *ROM*. A computer can read the contents of ROM, but it cannot change its contents, or store additional data there. ROM is *nonvolatile*, which means that it does not lose its contents, even when the computer's power is turned off. ROM is typically used to store programs that are important for the system's operation. One example is the computer's startup program, which is executed each time the computer is started.

Secondary Storage Devices

Secondary storage is a type of memory that can hold data for long periods of time, even when there is no power to the computer. Programs are normally stored in secondary memory and loaded into main memory as needed. Important data, such as word processing documents, payroll data, and inventory records, is saved to secondary storage as well.

The most common type of secondary storage device is the disk drive. A traditional *disk drive* stores data by magnetically encoding it onto a circular disk. *Solid state drives*, which store data in solid-state memory, are increasingly becoming popular. A solid state drive has no moving parts, and operates faster than a traditional disk drive. Most computers have some sort of secondary storage device, either a traditional disk drive or a solid state drive, mounted inside their case. External disk drives, which connect to one of the computer's communication ports, are also available. External disk drives can be used to create backup copies of important data or to move data to another computer.

In addition to external disk drives, many types of devices have been created for copying data, and for moving it to other computers. *Universal Serial Bus drives*, or *USB drives*, are small devices that plug into the computer's USB port, and appear to the system as a disk drive. These drives do not actually contain a disk, however. They store data in a special type of memory known as *flash memory*. USB drives, which are also known as *memory sticks* and *flash drives*, are inexpensive, reliable, and small enough to be carried in your pocket.



NOTE: In recent years, *cloud storage* has become a popular way to store data. When you store data in the cloud, you are storing it on a remote server via the Internet, or via a company's private network. When your data is stored in the cloud, you can access it from many different devices, and from any location where you have a network connection. Cloud storage can also be used to back up important data that is stored on a computer's disk.

Input Devices

Input is any data the computer collects from people and from other devices. The component that collects the data and sends it to the computer is called an *input device*. Common input devices are the keyboard, mouse, touchscreen, scanner, microphone, and digital camera. Disk drives and optical drives can also be considered input devices because programs and data are retrieved from them and loaded into the computer's memory.

Output Devices

Output is any data the computer produces for people or for other devices. It might be a sales report, a list of names, or a graphic image. The data is sent to an *output device*, which formats and presents it. Common output devices are video displays and printers. Disk drives can also be considered output devices because the system sends data to them in order to be saved.



Checkpoint

- 1.1 What is a program?
- 1.2 What is hardware?
- 1.3 List the five major components of a computer system.
- 1.4 What part of the computer actually runs programs?
- 1.5 What part of the computer serves as a work area to store a program and its data while the program is running?
- 1.6 What part of the computer holds data for long periods of time, even when there is no power to the computer?
- 1.7 What part of the computer collects data from people and from other devices?
- 1.8 What part of the computer formats and presents data for people or other devices?

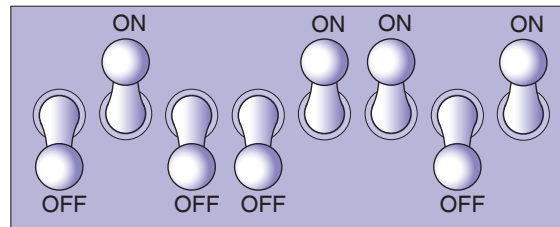
1.3 How Computers Store Data

CONCEPT: All data that is stored in a computer is converted to sequences of 0s and 1s.

A computer's memory is divided into tiny storage locations known as *bytes*. One byte is only enough memory to store a letter of the alphabet or a small number. In order to do anything meaningful, a computer has to have lots of bytes. Most computers today have millions, or even billions, of bytes of memory.

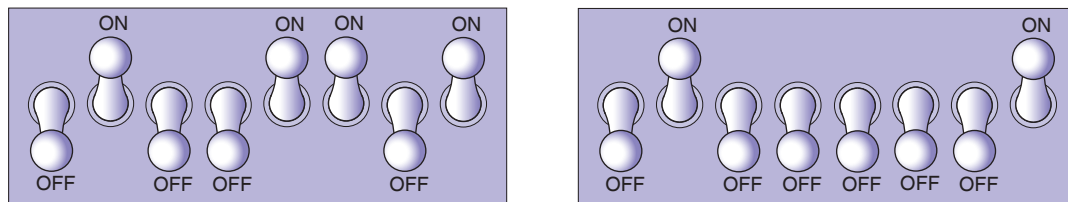
Each byte is divided into eight smaller storage locations known as bits. The term *bit* stands for *binary digit*. Computer scientists usually think of bits as tiny switches that can be either on or off. Bits aren't actual "switches," however, at least not in the conventional sense. In most computer systems, bits are tiny electrical components that can hold either a positive or a negative charge. Computer scientists think of a positive charge as a switch in the *on* position, and a negative charge as a switch in the *off* position. Figure 1-6 shows the way that a computer scientist might think of a byte of memory: as a collection of switches that are each flipped to either the on or off position.

Figure 1-6 Think of a byte as eight switches



When a piece of data is stored in a byte, the computer sets the eight bits to an on/off pattern that represents the data. For example, the pattern shown on the left in Figure 1-7 shows how the number 77 would be stored in a byte, and the pattern on the right shows how the letter A would be stored in a byte. In a moment you will see how these patterns are determined.

Figure 1-7 Bit patterns for the number 77 and the letter A



The number 77 stored in a byte.

The letter A stored in a byte.

Storing Numbers

A bit can be used in a very limited way to represent numbers. Depending on whether the bit is turned on or off, it can represent one of two different values. In computer systems, a bit that is turned off represents the number 0 and a bit that is turned on represents the number 1. This corresponds perfectly to the *binary numbering system*. In the binary numbering system (or *binary*, as it is usually called) all numeric values are written as sequences of 0s and 1s. Here is an example of a number that is written in binary:

10011101

The position of each digit in a binary number has a value assigned to it. Starting with the rightmost digit and moving left, the position values are 2^0 , 2^1 , 2^2 , 2^3 , and so forth, as shown in Figure 1-8. Figure 1-9 shows the same diagram with the position values calculated. Starting with the rightmost digit and moving left, the position values are 1, 2, 4, 8, and so forth.

Figure 1-8 The values of binary digits as powers of 2

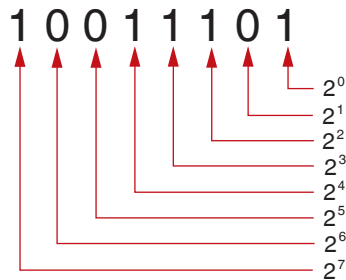
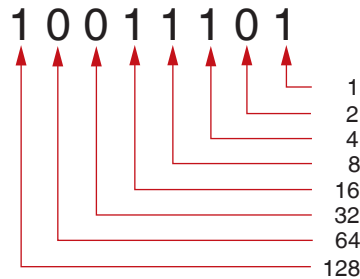


Figure 1-9 The values of binary digits



To determine the value of a binary number you simply add up the position values of all the 1s. For example, in the binary number 10011101, the position values of the 1s are 1, 4, 8, 16, and 128. This is shown in Figure 1-10. The sum of all of these position values is 157. So, the value of the binary number 10011101 is 157.

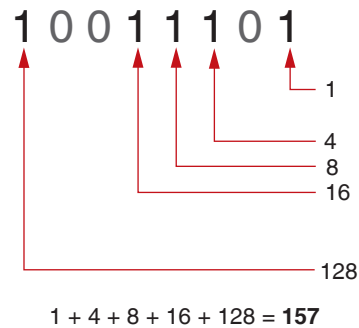
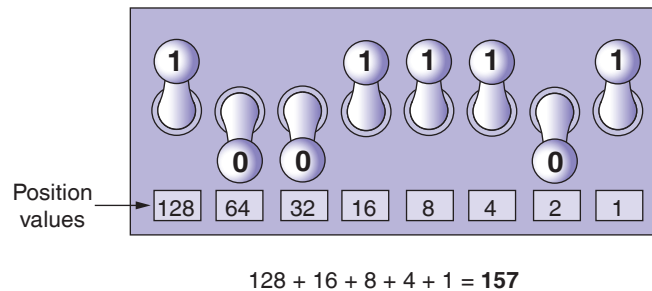
Figure 1-10 Determining the value of 10011101

Figure 1-11 shows how you can picture the number 157 stored in a byte of memory. Each 1 is represented by a bit in the on position, and each 0 is represented by a bit in the off position.

Figure 1-11 The bit pattern for 157

When all of the bits in a byte are set to 0 (turned off), then the value of the byte is 0. When all of the bits in a byte are set to 1 (turned on), then the byte holds the largest value that can be stored in it. The largest value that can be stored in a byte is $1 + 2 + 4 + 8 + 16 + 32 + 64 + 128 = 255$. This limit exists because there are only eight bits in a byte.

What if you need to store a number larger than 255? The answer is simple: use more than one byte. For example, suppose we put two bytes together. That gives us 16 bits. The position values of those 16 bits would be $2^0, 2^1, 2^2, 2^3$, and so forth, up through 2^{15} . As shown in Figure 1-12, the maximum value that can be stored in two bytes is 65,535. If you need to store a number larger than this, then more bytes are necessary.

Figure 1-12 Two bytes used for a large number