Preface

Welcome to *Starting Out with C++: From Control Structures through Objects, 7th edition*. This book is intended for use in a two-semester C++ programming sequence, or an accelerated one-semester course. Students new to programming, as well as those with prior course work in other languages, will find this text beneficial. The fundamentals of programming are covered for the novice, while the details, pitfalls, and nuances of the C++ language are explored in-depth for both the beginner and more experienced student. The book is written with clear, easy-to-understand language and it covers all the necessary topics for an introductory programming course. This text is rich in example programs that are concise, practical, and real-world oriented, ensuring that the student not only learns how to implement the features and constructs of C++, but why and when to use them.

Changes in the Seventh 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:

- This edition uses `string` objects, instead of `char` arrays, as the preferred way to store strings. This change has been made throughout the entire book. A thorough discussion of C-strings and the technique of storing them in `char` arrays is provided as a topic in Chapter 10.
- All of the introductory file I/O material has been consolidated and moved to Chapter 5. In previous editions, this material was gradually introduced in Chapters 3 through 5. Many reviewers requested that all the material be given in one place, after loops have been covered.
- Named constants are now introduced in Chapter 2, after variables.
- In Chapter 2 an additional *In the Spotlight* section demonstrating the modulus operator has been added.
- Chapter 4 has been reorganized so that all the fundamental decision structure topics appear early in the chapter.
- A discussion of passing arrays using `const` references has been added to Chapter 7.

- An *In the Spotlight* section giving an additional example of inheritance has been added to Chapter 15.
- Template examples for stacks, queues, and binary search trees have been added to Chapters 18 and 20.
- The Serendipity Booksellers project has been moved to the book's online resource page at www.pearsonhighered.com/gaddis.

Organization of the Text

This text teaches C++ in a step-by-step fashion. Each chapter covers a major set of topics and builds knowledge as the student progresses through the book. Although the chapters can be easily taught in their existing sequence, some flexibility is provided. The diagram shown in Figure P-1 suggests possible sequences of instruction.

Chapter 1 covers fundamental hardware, software, and programming concepts. You may choose to skip this chapter if the class has already mastered those topics. Chapters 2 through 7 cover basic C++ syntax, data types, expressions, selection structures, repetition structures, functions, and arrays. Each of these chapters builds on the previous chapter and should be covered in the order presented.

After Chapter 7 has been covered, you may proceed to Chapter 8, or jump to either Chapter 9 or Chapter 12. (If you jump to Chapter 12 at this point, you will need to postpone sections 12.7, 12.8, and 12.10 until Chapters 9 and 11 have been covered.)

After Chapter 9 has been covered, either of Chapters 10 or 11 may be covered. After Chapter 11, you may cover Chapters 13 through 17 in sequence. Next you can proceed to either Chapter 18 or Chapter 19. Finally, Chapter 20 may be covered.

This text's approach starts with a firm foundation in structured, procedural programming before delving fully into object-oriented programming and advanced data structures.

Brief Overview of Each Chapter

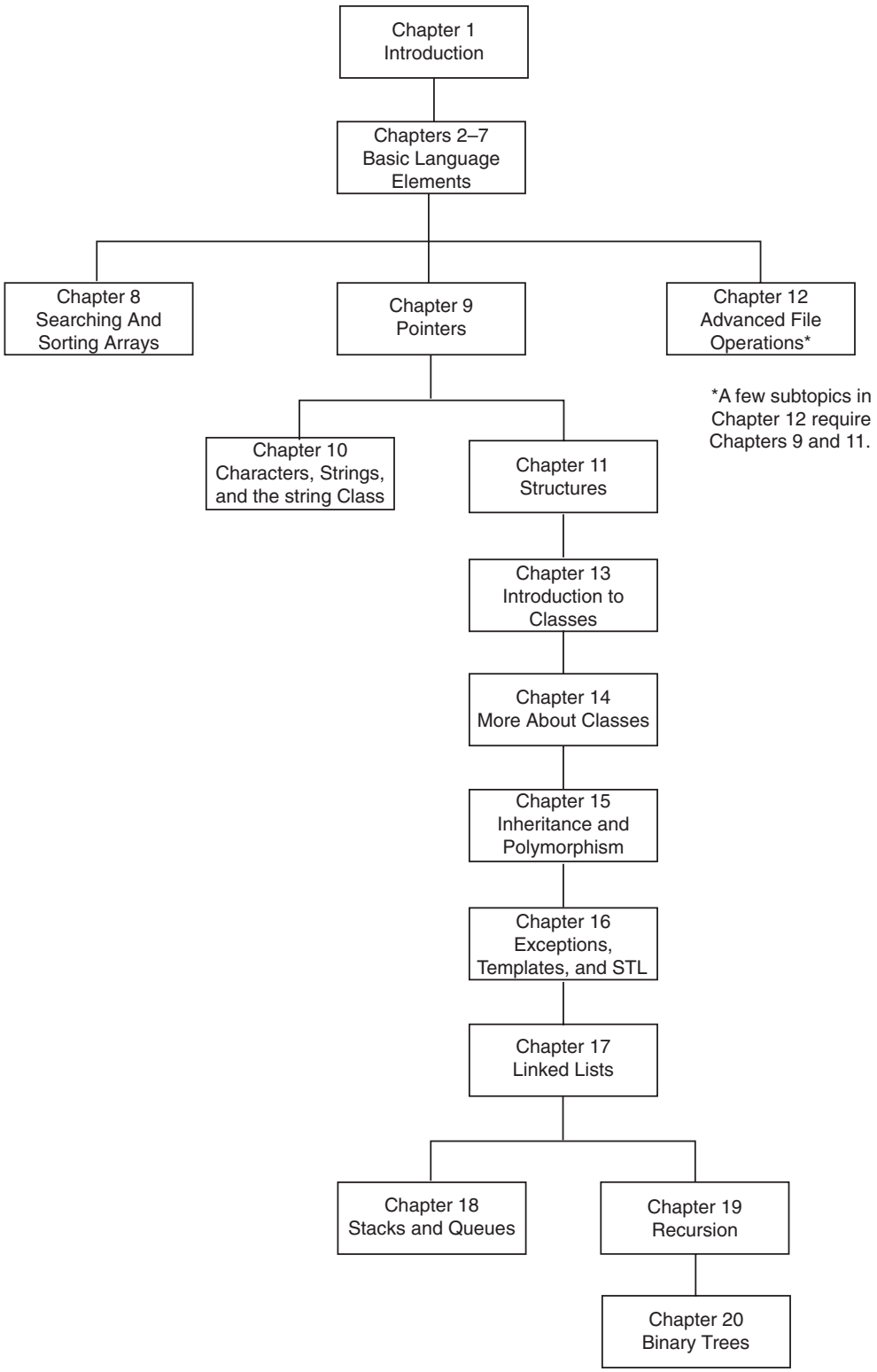
Chapter 1: Introduction to Computers and Programming

This chapter provides an introduction to the field of computer science and covers the fundamentals of programming, problem solving, and software design. The components of programs, such as key words, variables, operators, and punctuation are covered. The tools of the trade, such as pseudocode, flow charts, and hierarchy charts are also presented.

Chapter 2: Introduction to C++

This chapter gets the student started in C++ by introducing data types, identifiers, variable declarations, constants, comments, program output, simple arithmetic operations, and C-strings. Programming style conventions are introduced and good programming style is modeled here, as it is throughout the text. An optional section explains the difference between ANSI standard and pre-standard C++ programs.

Figure P-1



Chapter 3: Expressions and Interactivity

In this chapter the student learns to write programs that input and handle numeric, character, and string data. The use of arithmetic operators and the creation of mathematical expressions are covered in greater detail, with emphasis on operator precedence. Debugging is introduced, with a section on hand tracing a program. Sections are also included on simple output formatting, on data type conversion and type casting, and on using library functions that work with numbers.

Chapter 4: Making Decisions

Here the student learns about relational operators, relational expressions and how to control the flow of a program with the `if`, `if/else`, and `if/else if` statements. The conditional operator and the `switch` statement are also covered. Crucial applications of these constructs are covered, such as menu-driven programs and the validation of input.

Chapter 5: Loops and Files

This chapter covers repetition control structures. The `while` loop, `do-while` loop, and `for` loop are taught, along with common uses for these devices. Counters, accumulators, running totals, sentinels, and other application-related topics are discussed. Sequential file I/O is also introduced. The student learns to read and write text files, and use loops to process the data in a file.

Chapter 6: Functions

In this chapter the student learns how and why to modularize programs, using both `void` and value returning functions. Argument passing is covered, with emphasis on when arguments should be passed by value versus when they need to be passed by reference. Scope of variables is covered and sections are provided on local versus global variables and on static local variables. Overloaded functions are also introduced and demonstrated.

Chapter 7: Arrays

In this chapter the student learns to create and work with single and multidimensional arrays. Many examples of array processing are provided including examples illustrating how to find the sum, average, 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 and the student is shown how to use a data file as an input source to populate an array. STL vectors are introduced and compared to arrays.

Chapter 8: Sorting and Searching Arrays

Here the student learns the basics of sorting arrays and searching for data stored in them. The chapter covers the Bubble Sort, Selection Sort, Linear Search, and Binary Search algorithms. There is also a section on sorting and searching STL `vector` objects.

Chapter 9: Pointers

This chapter explains how to use pointers. Pointers are compared to and contrasted with reference variables. Other topics include pointer arithmetic, initialization of pointers, relational comparison of pointers, pointers and arrays, pointers and functions, dynamic memory allocation, and more.

Chapter 10: Characters, C-strings, and More About the `string` Class

This chapter discusses various ways to process text at a detailed level. Library functions for testing and manipulating characters are introduced. C-strings are discussed, and the technique of storing C-strings in `char` arrays is covered. An extensive discussion of the `string` class methods is also given.

Chapter 11: Structured Data

The student is introduced to abstract data types and taught how to create them using structures, unions, and enumerated data types. Discussions and examples include using pointers to structures, passing structures to functions, and returning structures from functions.

Chapter 12: Advanced File Operations

This chapter covers sequential access, random access, text, and binary files. The various modes for opening files are discussed, as well as the many methods for reading and writing file contents. Advanced output formatting is also covered.

Chapter 13: Introduction to Classes

The student now shifts focus to the object-oriented paradigm. This chapter covers the fundamental concepts of classes. Member variables and functions are discussed. The student learns about private and public access specifications, and reasons to use each. The topics of constructors, overloaded constructors, and destructors are also presented. The chapter presents a section modeling classes with UML, and how to find the classes in a particular problem.

Chapter 14: More About Classes

This chapter continues the study of classes. Static members, friends, memberwise assignment, and copy constructors are discussed. The chapter also includes in-depth sections on operator overloading, object conversion, and object aggregation. There is also a section on class collaborations and the use of CRC cards.

Chapter 15: Inheritance and Polymorphism

The study of classes continues in this chapter with the subjects of inheritance, polymorphism, and virtual member functions. The topics covered include base and derived class constructors and destructors, virtual member functions, base class pointers, static and dynamic binding, multiple inheritance, and class hierarchies.

Chapter 16: Exceptions, Templates, and the Standard Template Library (STL)

The student learns to develop enhanced error trapping techniques using exceptions. Discussion then turns to function and class templates as a method for reusing code. Finally, the student is introduced to the containers, iterators, and algorithms offered by the Standard Template Library (STL).

Chapter 17: Linked Lists

This chapter introduces concepts and techniques needed to work with lists. A linked list ADT is developed and the student is taught to code operations such as creating a linked list, appending a node, traversing the list, searching for a node, inserting a node, deleting a node, and destroying a list. A linked list class template is also demonstrated.

Chapter 18: Stacks and Queues

In this chapter the student learns to create and use static and dynamic stacks and queues. The operations of stacks and queues are defined, and templates for each ADT are demonstrated.

Chapter 19: Recursion

This chapter discusses recursion and its use in problem solving. A visual trace of recursive calls is provided and recursive applications are discussed. Many recursive algorithms are presented, including recursive functions for finding factorials, finding a greatest common denominator (GCD), performing a binary search, and sorting (QuickSort). The classic Towers of Hanoi example is also presented. For students who need more challenge, there is a section on exhaustive algorithms.

Chapter 20: Binary Trees

This chapter covers the binary tree ADT, and demonstrates many binary tree operations. The student learns to traverse a tree, insert an element, delete an element, replace an element, test for an element, and destroy a tree.

Appendix A: Getting Started with Alice

This appendix gives a quick introduction to Alice. Alice is free software that can be used to teach fundamental programming concepts using 3D graphics.

Appendix B: ASCII Character Set

A list of the ASCII and Extended ASCII characters and their codes.

Appendix C: Operator Precedence and Associativity

A chart showing the C++ operators and their precedence.

The following appendices are available online at www.pearsonhighered.com/gaddis.

Appendix D: Introduction to Flowcharting

A brief introduction to flowcharting. This tutorial discusses sequence, selection, case, repetition, and module structures.

Appendix E: Using UML in Class Design

This appendix shows the student how to use the Unified Modeling Language to design classes. Notation for showing access specification, data types, parameters, return values, overloaded functions, composition, and inheritance are included.

Appendix F: Namespaces

This appendix explains namespaces and their purpose. Examples showing how to define a namespace and access its members are given.

Appendix G: Writing Managed C++ Code for the .NET Framework

This appendix introduces the student to the concepts surrounding managed C++ in Microsoft's .NET environment.

Appendix H: Passing Command Line Arguments

Teaches the student how to write a C++ program that accepts arguments from the command line. This appendix will be useful to students working in a command line environment, such as Unix, Linux, or the Windows command prompt.

Appendix I: Header File and Library Function Reference

This appendix provides a reference for the C++ library functions and header files discussed in the book.

Appendix J: Binary Numbers and Bitwise Operations

A guide to the C++ bitwise operators, as well as a tutorial on the internal storage of integers.

Appendix K: Multi-Source File Programs

Provides a tutorial on creating programs that consist of multiple source files. Function header files, class specification files, and class implementation files are discussed.

Appendix L: Stream Member Functions for Formatting

Covers stream member functions for formatting such as `setf`.

Appendix M: Introduction to Microsoft Visual C++ 2010 Express Edition

A tutorial on how to start a project in Microsoft Visual C++ 2010 Express Edition, compile a program, save source files, and more.

Appendix N: Answers to Checkpoints

Students may test their own progress by comparing their answers to the checkpoint exercises against this appendix. The answers to all Checkpoints are included.

Appendix O: Solutions to Odd-Numbered Review Questions

Another tool that students can use to gauge their progress.

Features of the Text

Concept Statements	Each major section of the text starts with a concept statement. This statement summarizes the ideas of the section.
Example Programs	The text has hundreds of complete example programs, each designed to highlight the topic currently being studied. In most cases, these are practical, real-world examples. Source code for these programs is provided so that students can run the programs themselves.
Program Output	After each example program there is a sample of its screen output. This immediately shows the student how the program should function.
 **In the Spotlight**	Each of these sections provides a programming problem and a detailed, step by step analysis showing the student how to solve it.
 **VideoNotes**	A series of online videos, developed specifically for this book, is available for viewing at `www.pearsonhighered.com/gaddis`. Icons appear throughout the text alerting the student to videos about specific topics.
 **Checkpoints**	Checkpoints are questions placed throughout each chapter as a self-test study aid. Answers for all Checkpoint questions can be downloaded from the book's Companion Website at www.pearsonhighered.com/gaddis. This allows students to check how well they have learned a new topic.
 **Notes**	Notes appear at appropriate places throughout the text. They are short explanations of interesting or often misunderstood points relevant to the topic at hand.
 **Warnings**	Warnings are notes that caution the student about certain C++ features, programming techniques, or practices that can lead to malfunctioning programs or lost data.

Case Studies Case studies that simulate real-world applications appear in many chapters throughout the text. These case studies are designed to highlight the major topics of the chapter in which they appear.

Review Questions and Exercises Each chapter presents a thorough and diverse set of review questions, such as fill-in-the-blank and short answer, that check the student's mastery of the basic material presented in the chapter. These are followed by exercises requiring problem solving and analysis, such as the *Algorithm Workbench*, *Predict the Output*, and *Find the Errors* sections. Answers to the odd numbered review questions and review exercises can be downloaded from the book's Companion Website at www.pearsonhighered.com/gaddis.

Programming Challenges Each chapter offers a pool of programming exercises designed to solidify the student's knowledge of the topics currently being studied. In most cases the assignments present real-world problems to be solved. When applicable, these exercises include input validation rules.

Group Projects There are several group programming projects throughout the text, intended to be constructed by a team of students. One student might build the program's user interface, while another student writes the mathematical code, and another designs and implements a class the program uses. This process is similar to the way many professional programs are written and encourages team work within the classroom.

Software Development Project: Serendipity Booksellers Available for download from the book's Companion Website at www.pearsonhighered.com/gaddis. This is an on-going project that instructors can optionally assign to teams of students. It systematically develops a "real-world" software package: a point-of-sale program for the fictitious Serendipity Booksellers organization. The Serendipity assignment for each chapter adds more functionality to the software, using constructs and techniques covered in that chapter. When complete, the program will act as a cash register, manage an inventory database, and produce a variety of reports.

C++ Quick Reference Guide For easy access, a quick reference guide to the C++ language is printed on the last two pages of Appendix C in the book.

Supplements

Student Online Resources

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

- The source code for each example program in the book
- Access to the book's companion VideoNotes
- A full set of appendices, including answers to the Checkpoint questions, and answers to the odd-numbered review questions
- A collection of valuable Case Studies
- The complete Serendipity Booksellers Project

Integrated Development Environment (IDE) Resource Kits

Professors who adopt this text can order it for students with a kit containing five popular C++ IDEs (Microsoft® Visual Studio 2010 Express Edition, Dev C++, NetBeans, Eclipse, and CodeLite) and access to a Web site containing written and video tutorials for getting started in each IDE. For ordering information, please contact your campus Pearson Education representative or visit www.pearsonhighered.com/cs.

myprogramminglab

Online Practice and Assessment with MyProgrammingLab

MyProgrammingLab helps students fully grasp the logic, semantics, and syntax of programming. Through practice exercises and immediate, personalized feedback, MyProgrammingLab improves the programming competence of beginning students who often struggle with the basic concepts and paradigms of popular high-level programming languages.

A self-study and homework tool, a MyProgrammingLab course consists of hundreds of small practice problems organized around the structure of this textbook. For students, the system automatically detects errors in the logic and syntax of their code submissions and offers targeted hints that enable students to figure out what went wrong—and why. For instructors, a comprehensive gradebook tracks correct and incorrect answers and stores the code inputted by students for review.

MyProgrammingLab is offered to users of this book in partnership with Turing's Craft, the makers of the CodeLab interactive programming exercise system. For a full demonstration, to see feedback from instructors and students, or to get started using MyProgrammingLab in your course, visit www.myprogramminglab.com.

Instructor Resources

The following supplements are available to qualified instructors only:

- Answers to all Review Questions in the text
- Solutions for all Programming Challenges in the text
- PowerPoint presentation slides for every chapter

- Computerized test bank
- Answers to all Student Lab Manual questions
- Solutions for all Student Lab Manual programs

Visit the Pearson Instructor Resource Center (`www.pearsonhighered.com/irc`) or send an email to `computing@pearson.com` for information on how to access them.

Textbook Web site

Student and instructor resources, including links to download Microsoft® Visual C++ 2010 Express and other popular IDEs, for all the books in the Gaddis *Starting Out With* series can be accessed at the following URL:

`http://www.pearsonhighered.com/gaddis`

Get this book the way you want it!

This book is part of Pearson Education's custom database for Computer Science textbooks. Use our online PubSelect system to select just the chapters you need from this, and other, Pearson Education CS textbooks. You can edit the sequence to exactly match your course organization and teaching approach. Visit `www.pearsoncustom.com/cs` for details.

Which Gaddis C++ book is right for you?

The Starting Out with C++ Series includes three books, one of which is sure to fit your course:

- *Starting Out with C++: From Control Structures through Objects*
- *Starting Out with C++: Early Objects*
- *Starting Out with C++: Brief Version.*

The following chart will help you determine which book is right for your course.

▪ FROM CONTROL STRUCTURES THROUGH OBJECTS ▪ BRIEF VERSION	▪ EARLY OBJECTS
LATE INTRODUCTION OF OBJECTS Classes are introduced in Chapter 13 of the standard text and Chapter 11 of the brief text, after control structures, functions, arrays, and pointers. Advanced OOP topics, such as inheritance and polymorphism, are covered in the following two chapters.	**EARLIER INTRODUCTION OF OBJECTS** Classes are introduced in Chapter 7, after control structures and functions, but before arrays and pointers. Their use is then integrated into the remainder of the text. Advanced OOP topics, such as inheritance and polymorphism, are covered in Chapters 11 and 15.
INTRODUCTION OF DATA STRUCTURES AND RECURSION Linked lists, stacks and queues, and binary trees are introduced in the final chapters of the standard text. Recursion is covered after stacks and queues, but before binary trees. These topics are not covered in the brief text, though it does have appendices dealing with linked lists and recursion.	**INTRODUCTION OF DATA STRUCTURES AND RECURSION** Linked lists, stacks and queues, and binary trees are introduced in the final chapters of the text, after the chapter on recursion.

Acknowledgments

There have been many helping hands in the development and publication of this text. We would like to thank the following faculty reviewers for their helpful suggestions and expertise.

Ahmad Abuhejleh
University of Wisconsin, River Falls

David Akins
El Camino College

Steve Allan
Utah State University

Vicki Allan
Utah State University

Karen M. Arlien
Bismark State College

Mary Astone
Troy University

Ijaz A. Awan
Savannah State University

Robert Baird
Salt Lake Community College

Don Biggerstaff
Fayetteville Technical Community College

Michael Bolton
Northeastern Oklahoma State University

Bill Brown
Pikes Peak Community College

Charles Cadenhead
Richland Community College

Randall Campbell
Morningside College

Wayne Caruolo
Red Rocks Community College

Cathi Chambley-Miller
Aiken Technical College

C.C. Chao
Jacksonville State University

Joseph Chao
Bowling Green State University

Royce Curtis
Western Wisconsin Technical College

Joseph DeLibero
Arizona State University

Jeanne Douglas
University of Vermont

Michael Dowell
Augusta State U

William E. Duncan
Louisiana State University

Judy Etchison
Southern Methodist University

Dennis Fairclough
Utah Valley State College

Mark Fienup
University of Northern Iowa

Richard Flint
North Central College

Ann Ford Tyson
Florida State University

Jeanette Gibbons
South Dakota State University

James Gifford
University of Wisconsin, Stevens Point

Leon Gleiberman
Touro College

Barbara Guillott
Louisiana State University

Ranette Halverson, Ph.D.
Midwestern State University

Carol Hannahs
University of Kentucky

Dennis Heckman
Portland Community College

Ric Heishman
George Mason University

Michael Hennessy
University of Oregon

Ilga Higbee
Black Hawk College

Patricia Hines
Brookdale Community College

Mike Holland
Northern Virginia Community College

Mary Hovik
Lehigh Carbon Community College

Richard Hull
Lenoir-Rhyne College

Chris Kardaras
North Central College

Willard Keeling
Blue Ridge Community College

A.J. Krygeris
Houston Community College

Sheila Lancaster
Gadsden State Community College

Ray Larson
Inver Hills Community College

Jennifer Li
Ohlone College

Norman H. Liebling
San Jacinto College

Zhu-qu Lu
University of Maine, Presque Isle

Heidar Malki
University of Houston

Debbie Mathews
J. Sargeant Reynolds

Rick Matzen
Northeastern State University

Robert McDonald
East Stroudsburg University

James McGuffee
Austin Community College

Dean Mellas
Cerritos College

Lisa Milkowski
Milwaukee School of Engineering

Marguerite Nedreberg
Youngstown State University

Lynne O'Hanlon
Los Angeles Pierce College

Frank Paiano
Southwestern Community College

Theresa Park
Texas State Technical College

Mark Parker
Shoreline Community College

Tino Posillico
SUNY Farmingdale

Frederick Pratter
Eastern Oregon University

Susan L. Quick
Penn State University

Alberto Ramon
Diablo Valley College

Bazlur Rasheed
Sault College of Applied Arts and Technology

Farshad Ravanshad
Bergen Community College

Dolly Samson
Weber State University

Ruth Sapir
SUNY Farmingdale

Jason Schatz
City College of San Francisco

Dr. Sung Shin
South Dakota State University

Bari Siddique
University of Texas at Brownsville

William Slater
Collin County Community College

Shep Smithline
University of Minnesota

Caroline St. Claire
North Central College

Kirk Stephens
Southwestern Community College

Cherie Stevens
South Florida Community College

Dale Suggs
Campbell University

Mark Swanson
Red Wing Technical College

Ann Sudell Thorn
Del Mar College

Martha Tillman
College of San Mateo

Ralph Tomlinson
Iowa State University

David Topham
Ohlone College

Robert Tureman
Paul D. Camp Community College

Arisa K. Ude
Richland College

Peter van der Goes
Rose State College

Stewart Venit
California State University, Los Angeles

Judy Walters
North Central College

John H. Whipple
Northampton Community College

Aurelia Williams
Norfolk State University

Vida Winans
Illinois Institute of Technology

I would like to thank my family for their love and support in all of my many projects. I would also like to thank Christopher Rich for his assistance in this revision. I am extremely fortunate to have Michael Hirsch as my editor, and Stephanie Sellinger as editorial assistant. Michael's support and encouragement makes it a pleasure to write chapters and meet deadlines. I am also fortunate to have Yez Alayan as marketing manager, and Kathryn Ferranti as marketing coordinator. They do a great job getting my books out to the academic community. I had a great production team led by Jeff Holcomb, Managing Editor, and Marilyn Lloyd, Senior Production Project Manager. Thanks to you all!

About the Author

Tony Gaddis is the principal author of the *Starting Out with* series of textbooks. He has nearly two decades of experience teaching computer science courses, primarily at Haywood Community College. Tony 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 textbooks covering Programming Logic and Design, Alice, C++, Java™, Microsoft® Visual Basic®, Microsoft® Visual C#, and Python, all published by Pearson Addison-Wesley.