Tenth Edition

Starting Out with

C++ Early Objects

Tony Gaddis
Judy Walters
Godfrey Muganda

Senior Vice President Courseware Portfolio Management: Marcia Horton
Vice President Courseware Engineering, Computer Science & Global Editions: Julian Partridge
Executive Portfolio Manager: Matt Goldstein
Portfolio Management Assistant: Meghan Jacoby
Product Marketing Manager: Yvonne Vannatta
Field Marketing Manager: Demetrius Hall
Marketing Assistant: Jon Bryant
Managing Producer: Scott Disanno
Content Producer: Amanda Brands
Manufacturing Buyer, Higher Ed, Lake Side Communications, Inc. (LSC): Maura Zaldivar-Garcia
Cover Designer: Pearson CSC
Cover: bluestocking/Getty Images
Manager, Rights and Permissions: Ben Ferrini
Inventory Manager: Bruce Boundy
Full-Service Project Management: Abhishan Sharma, Integra Software Services Pvt. Ltd.
Composition: Integra Software Services Pvt. Ltd.
Printer/Binder: Lake Side Communications, Inc.

Library of Congress Cataloging-in-Publication Data

Names: Gaddis, Tony, author. | Walters, Judy, author. | Muganda, Godfrey, author.
Title: Starting out with C++. Early objects / Tony Gaddis, Judy Walters, Godfrey Muganda.
Description: Tenth edition. | Hoboken : Pearson Education, Inc., [2020] | Includes index.
Identifiers: LCCN 2018048221| ISBN 9780135235003 | ISBN 0135235006
Subjects: LCSH: C++ (Computer program language)
Classification: LCC QA76.73.C153 G333 2020 | DDC 005.13/3—dc23 LC record available at https://lccn.loc.gov/2018048221

10 9 8 7 6 5 4 3 2 1

Pearson

ISBN 10: 0-13-523500-6
ISBN 13: 978-0-13-523500-3

Contents at a Glance

Contents

Additional Appendices

The following appendices are located on the book's companion web site.

LOCATION OF VIDEONOTES IN THE TEXT

VideoNote

Chapter 1	Designing a Program with Pseudocode, p. 20 Designing the Account Balance Program, p. 25 Predicting the Output of Problem 33, p. 26 Solving the Candy Bar Sales Problem, p. 27
Chapter 2	Using `cout` to Display Output, p. 34 Assignment Statements, p. 61 Arithmetic Operators, p. 64 Solving the Restaurant Bill Problem, p. 76
Chapter 3	Using `cin` to Read Input, p. 79 Evaluating Mathematical Expressions, p. 86 Combined Assignment Operators, p. 104 Solving the Stadium Seating Problem, p. 150
Chapter 4	Using an `if` Statement, p. 163 Using an `if/else` Statement, p. 172 Using an `if/else if` Statement, p. 178 Using Logical Operators, p. 191 Solving the Time Calculator Problem, p. 241
Chapter 5	The `while` Loop, p. 248 The `for` Loop, p. 275 Nested Loops, p. 283 Solving the Ocean Levels Problem, p. 320
Chapter 6	Defining and Calling Functions, p. 328 Using Function Arguments, p. 337 Value-Returning Functions, p. 347 Solving the Markup Problem, p. 404
Chapter 7	Creating a Class, p. 416 Creating and Using Class Objects, p. 418 Creating and Using Structures, p. 460 Solving the `Car` Class Problem, p. 507
Chapter 8	Accessing Array Elements, p. 515 Passing an Array to a Function, p. 550 Two-Dimensional Arrays, p. 560 Solving the Chips and Salsa Problem, p. 603
Chapter 9	Performing a Binary Search, p. 616 Sorting a Set of Data, p. 623 Solving the Lottery Winners Problem, p. 655

(continued on next page)

LOCATION OF VIDEONOTES IN THE TEXT *(continued)*

VideoNote

Preface

Welcome to *Starting Out with C++: Early Objects,* 10th Edition. This book is intended for use in a two-term or three-term C++ programming sequence, or an accelerated one-term 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.

What's New in the Tenth Edition

While this book's pedagogy, organization, and clear writing style remain the same as in the previous edition, many updates and improvements have been made throughout the text. Here is a summary of some of the major changes.

- Additional features of the C++11 standard have been included.
  - The C++11 standard was a major revision of the C++ language that added many new features. We introduced some of these in the ninth edition of this text. This edition extends that coverage, introducing additional features.
  - Almost all newer C++ compilers support the C++11 standard, and we expect most students will be using one of these. However, the book can be used with an older compiler. As you progress through the chapters, you will see C++11 icons in the margins next to material on features new to C++11. Programs appearing in sections that are not marked with this icon will still compile using an older compiler.
- New or revised material has been included on a number of topics including alternate forms of variable initialization, Boolean expressions and variables, character conversion and testing, string processing, searching and sorting, vectors, pointers, class member initialization lists, and constructor delegation.

- The material on the Standard Template Library (STL) has been moved to its own chapter and rewritten with expanded material.
- The bubble sort algorithm, presented in Chapter 9, has been completely rewritten to be simpler for students to understand. It is followed by new material on how to modify the algorithm to increase its efficiency. Thirteen new figures have been added to the chapter to illustrate step-by-step how both the bubble sort and selection sort work.
- Many additional figures throughout the book have been improved and some new ones added to help students visualize additional important concepts.
- Many new and updated programs, checkpoint questions, end-of-chapter questions and exercises, and programming challenge problems have been added throughout the book.

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, flexibility is provided. The dependency diagram on the following page (Figure P-1) suggests possible sequences of instruction.

Chapter 1 covers fundamental hardware, software, and programming concepts. The instructor may choose to skip this chapter if the class has already mastered those topics.

Chapters 2 through 6 cover basic C++ syntax, data types, expressions, selection structures, repetition structures, and functions. Each of these chapters builds on the previous chapter and should be covered in the order presented.

Chapter 7 introduces object-oriented programming. It can be covered any time after Chapter 6, but before Chapter 11.

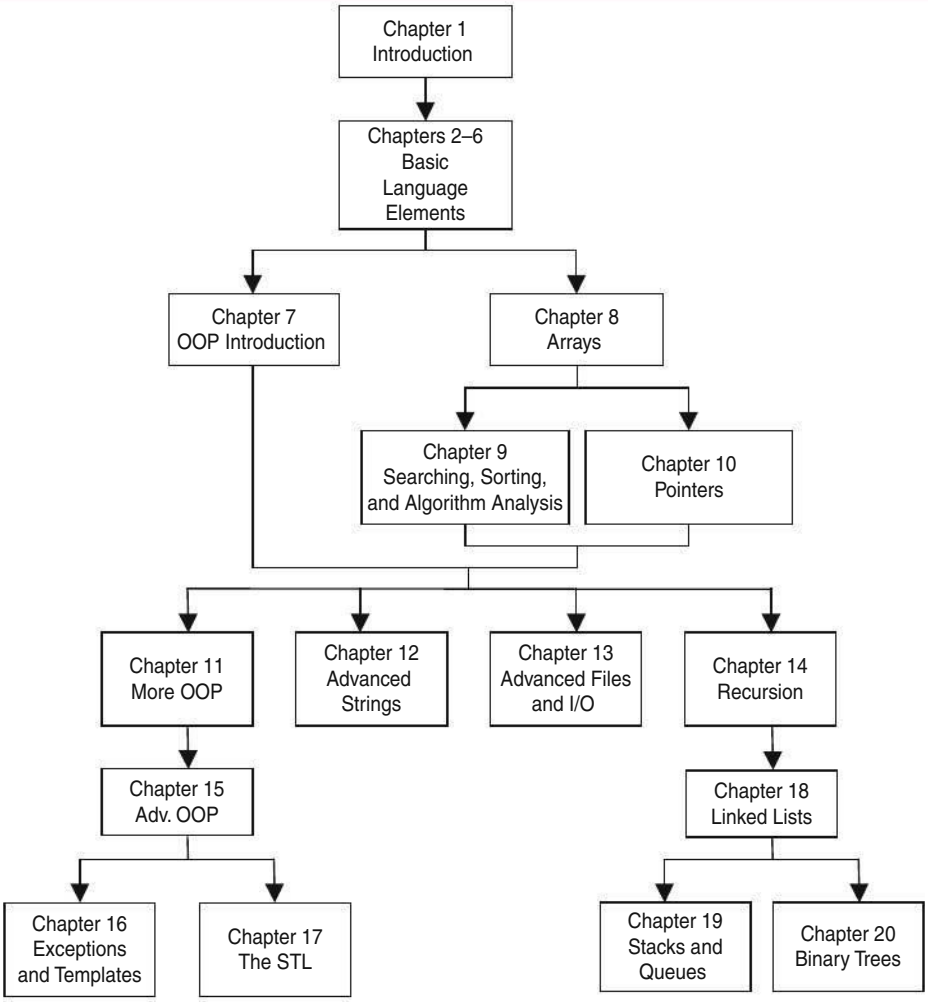
Instructors who prefer to introduce arrays before classes can cover Chapter 8 before Chapter 7. In this case it is only necessary to postpone Section 8.13 (Arrays of Objects) until Chapter 7 has been covered.

As Figure P-1 illustrates, in the second half of the book Chapters 11, 12, 13, and 14 can be covered in any order. Chapters 11, 15, and 16, however, should be done in sequence.

Chapter 17 (The Standard Template Library) can be covered any time after Chapter 15, although some instructors prefer to cover it after Chapter 16 (Exceptions and Templates).

Chapters 18-20 (Data structures) can be taught at several different points in the course. Some instructors prefer to wait and cover this material after Chapters 16 and 17 on templates and the STL. However, instructors who wish to introduce data structures at an earlier point in the course can cover them any time after Chapter 14 (Recursion) by simply omitting sections that deal with templates and the Standard Template Library.

Figure P-1


Chapter 1
Introduction
Chapters 2–6
Basic
Language
Elements
Chapter 7
OOP Introduction
Chapter 8
Arrays
Chapter 9
Searching, Sorting,
and Algorithm Analysis
Chapter 10
Pointers
Chapter 11
More OOP
Chapter 12
Advanced
Strings
Chapter 13
Advanced Files
and I/O
Chapter 14
Recursion
Chapter 15
Adv. OOP
Chapter 18
Linked Lists
Chapter 16
Exceptions
and Templates
Chapter 17
The STL
Chapter 19
Stacks and
Queues
Chapter 20
Binary Trees

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 hardware, software, operating systems, programming, problem solving, and software engineering. The components of programs, such as key words, variables, operators, and punctuation are covered. The tools of the trade, such as hierarchy charts and pseudocode, are also presented. The *Tying It All Together* section shows students how to use the `cout` statement to create a personalized output message. Programming Challenges at the end of the chapter help students see how the same basic input, processing, and output structure can be used to create multiple programs.

Chapter 2: Introduction to C++

This chapter gets the student started in C++ by introducing the basic parts of a C++ program, data types, the use of variables and literals, assignment statements, simple arithmetic operations, program output, and comments. The C++ `string` class is presented and `string` objects are used from this point on in the book as the primary method of handling strings. Programming style conventions are introduced, and good programming style is modeled here, as it is throughout the text. The *Tying It All Together* section lets the student play with simple text-based graphics.

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, with emphasis on operator precedence. Multiple assignment and combined assignment operators are also presented. Debugging is introduced, with a section on hand tracing a program. Additional sections cover using random numbers, simple output formatting, data type conversion and type casting, and library functions that work with numbers. The *Tying It All Together* section shows students how to create a simple interactive word game.

Chapter 4: Making Decisions

Here the student learns about relational expressions and how to control the flow of a program with `if`, `if/else`, and `if/else if` statements. Logical operators, the conditional operator, and the `switch` statement are also covered. Applications of these constructs, such as menu-driven programs, are illustrated. This chapter also introduces enumerated data types and the concepts of blocks and scope. It continues the theme of debugging with a section on validating output results. The *Tying It All Together* section uses random numbers and branching statements to create a fortune telling game.

Chapter 5: Looping

This chapter introduces C++'s repetitive control mechanisms. The `while` loop, `do-while` loop, and `for` loop are presented, along with a variety of methods to control them. These include using counters, user input, end sentinels, and end-of-file testing. Applications utilizing loops, such as keeping a running total and performing data validation, are also covered. The chapter includes an extensive section on working with files and a section on creating good test data, continuing the book's emphasis on testing and debugging. A new Programming Challenge shows students how to use C++ code to generate a simple .html web page, and The *Tying It All Together* section introduces students to Windows commands to create colorful output and use a loop to create a multi-colored display.

Chapter 6: Functions

In this chapter the student learns how and why to modularize programs, using both `void` and value-returning functions. Parameter 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. The *Tying It All Together* section includes a modular, menu-driven program that emphasizes the versatility of functions, illustrating how their behavior can be controlled by the arguments sent to them.

Chapter 7: Introduction to Classes and Objects

In this chapter the text begins to focus on the object-oriented paradigm. Students have used provided C++ classes since the beginning of the text, but now they learn how to define their own classes and to create and use objects of these classes. Careful attention is paid to illustrating which functions belong in a class versus which functions belong in a client program that uses the class. In addition to demonstrating how to create and use constructors, students are introduced to member initialization lists, in-place member initialization, and constructor delegation. Good object-oriented practices are discussed and modeled, such as protecting member data through carefully constructed accessor and mutator functions and hiding class implementation details from client programs. Once students are comfortable working with classes and objects, the chapter introduces object composition. It also includes a brief introduction to the topic of object-oriented analysis and design and sections on structures and on screen control techniques, both of which are used in the *Tying It All Together* section where students create a yoyo animation.

Chapter 8: Arrays

In this chapter the student learns to create and work with single and multidimensional arrays. Many examples of array processing are provided, including functions to compute the sum, average, highest and lowest values in an array. Students also learn to create tables using two-dimensional arrays, and to analyze array data by row or by column. 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. The range-based `for` loop is introduced as an easy way to iterate through all the elements of an array, and STL vectors are introduced and compared to arrays. A section on arrays of objects and structures is located at the end of the chapter, so it can be covered now or saved for later if the instructor wishes to cover this chapter before Chapter 7. The *Tying It All Together* section uses arrays to create a game of *Rock*, *Paper*, *Scissors* between a human player and the computer.

Chapter 9: Searching, Sorting, and Algorithm Analysis

Here the student learns the basics of searching for information stored in arrays and of sorting arrays, including arrays of objects. The chapter covers the Linear Search, Binary Search, Bubble Sort, and Selection Sort algorithms and has an optional section on sorting and searching STL vectors. A brief introduction to algorithm analysis is included, and students are shown how to determine which of two algorithms is more efficient. This chapter's *Tying It All Together* section uses both a table lookup and a searching algorithm to encode and decode secret messages.

Chapter 10: Pointers

This chapter explains how to use pointers. Topics include pointer arithmetic, initialization of pointers, comparison of pointers, pointers and arrays, pointers and functions, dynamic memory allocation, the `nullptr` key word, and more. A section introducing smart pointers focuses on `shared_ptrs` and `unique_ptrs`, and shows how they can be used to avoid memory leaks. The *Tying It All Together* section demonstrates the use of pointers to access library data structures and functions that return calendar and wall clock time.

Chapter 11: More About Classes and Object-Oriented Programming

This chapter continues the study of classes and object-oriented programming, covering more advanced topics such as inheritance and object aggregation and composition. Other topics include the `this` pointer, constant member functions, static members, friends, memberwise assignment, copy constructors, object type conversion operators, convert constructors, operator overloading, move constructors, move assignment operators, and overriding base class functions. New figures have been added to illustrate and clarify the concepts of aggregation and composition. The *Tying It All Together* section brings together the concepts of inheritance and convert constructors to build a program that formats the contents of an array to form an HTML table for display on a Web site.

Chapter 12: More on C-Strings and the `string` Class

This chapter covers standard library functions for working with characters and C-strings, as well as material on using `string` objects. It includes sections on character testing and character conversion functions, `string` class functions, functions in the C++11 `string` library, and overloaded `to_string` functions for converting numeric values to `string` objects. The *Tying It All Together* section shows students how to access string-based program environments to obtain information about the computer and the network on which the program is running.

Chapter 13: Advanced File and I/O Operations

This chapter introduces more advanced topics for working with sequential access text files and introduces random access and binary files. Various modes for opening files are discussed, as well as the many methods for reading and writing their contents. The *Tying It All Together* program applies many of the techniques covered in the chapter to merge two text files into an HTML document for display on the Web, with different colors used to illustrate which file each piece of data came from.

Chapter 14: Recursion

In this chapter recursion is defined and demonstrated. A visual trace of recursive calls is provided, and recursive applications are discussed. Many recursive algorithms are presented, including recursive functions for computing factorials, finding a greatest common denominator (GCD), performing a binary search, sorting using QuickSort, and solving the famous Towers of Hanoi problem. For students who need more challenge, there is a section on exhaustive and enumeration algorithms. The *Tying It All Together* section uses recursion to evaluate prefix expressions.

Chapter 15: Polymorphism and Virtual Functions

The study of classes and object-oriented programming continues in this chapter with the introduction of more advanced concepts such as polymorphism and virtual functions. Information is also presented on abstract base classes, pure virtual functions, type compatibility within an inheritance hierarchy, and virtual inheritance. The *Tying It All Together* section illustrates the use of inheritance and polymorphism to display and animate graphical images.

Chapter 16: Exceptions and Templates

Here the student learns to develop enhanced error trapping techniques using exceptions. Discussion then turns to using function and class templates to create generic code.

Chapter 17: The Standard Template Library

This new chapter extends the STL material previously found in Chapter 16 to offer a comprehensive treatment of the containers, adapters, iterators, and algorithms that comprise the Standard Template Library (STL). It includes the `vector` class, the `map`, `multimap`, and `unordered_map` classes, and the `set`, `multiset`, and `unordered_set` classes. The chapter also introduces function objects and lambda expressions, and shows how to use them with STL algorithms. Many example programs are included to aid student understanding and many new checkpoints, review exercises, and programming challenges have been added to help students test their knowledge of concepts. The *Tying It All Together* section uses various containers in the Standard Template Library to create an educational children's game.

Chapter 18: Linked Lists

This chapter introduces concepts and techniques needed to work with lists. A linked list ADT is developed, and the student learns how to create and destroy a list, as well as to write functions to insert, append, and delete nodes, to traverse the list, and to search for a specific node. A linked list class template is demonstrated, the section on the STL `list` container has been rewritten, and information on the C++ 11 standard `forward_list` container has been added. The *Tying It All Together* section brings together many of the most important concepts of OOP by using objects, inheritance, and polymorphism in conjunction with the STL list class to animate a collection of images.

Chapter 19: 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. The static array-based stack uses exception-handling to handle stack overflow and underflow, providing a realistic and natural example of defining, throwing, and catching exceptions. The *Tying It All Together* section discusses strategies for evaluating postfix expressions and uses a stack to convert a postfix expression to infix.

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, delete, and replace elements, search for a particular element, and destroy a tree. The *Tying It All Together* section introduces a tree structure versatile enough to create genealogy trees.

Appendices in the Book

Appendix A: The ASCII Character Set A list of the ASCII and extended ASCII characters and their codes.

Appendix B: Operator Precedence and Associativity A list of the C++ operators with their precedence and associativity.

Appendix C: Answers to Checkpoints A tool students can use to assess their understanding by comparing their answers to the Checkpoint exercises found throughout the book. The answers to all Checkpoint exercises are included.

Appendix D: Answers to Odd-Numbered Review Questions Another tool students can use to gauge their understanding and progress.

Additional Appendices on the Book's Companion Website

Appendix E: A Brief Introduction to Object-Oriented Programming An introduction to the concepts and terminology of object-oriented programming.

Appendix F: Using UML in Class Design A brief introduction to the Unified Modeling Language (UML) class diagrams with examples of their use.

Appendix G: Multi-Source File Programs A tutorial on how to create, compile, and link programs with multiple source files. Includes the use of function header files, class specification files, and class implementation files.

Appendix H: Multiple and Virtual Inheritance A self-contained discussion of the C++ concepts of multiple and virtual inheritance for anyone already familiar with single inheritance.

Appendix I: Header File and Library Function Reference A reference for the C++ library functions and header files used in the book.

Appendix J: Namespaces An explanation of namespaces and their purpose, with examples provided on how to define a namespace and access its members.

Appendix K: C++ Casts and Run-Time Type Identification An introduction to different ways of doing type casting in C++ and to run-time type identification.

Appendix L: Passing Command Line Arguments An introduction to writing C++ programs that accept command-line arguments. This appendix will be useful to students working in a command-line environment, such as UNIX or Linux.

Appendix M: Binary Numbers and Bitwise Operations A guide to the binary number system and the C++ bitwise operators, as well as a tutorial on the internal storage of integers.

Appendix N: Introduction to Flowcharting A tutorial that introduces flowcharting and its symbols. It includes handling sequence, selection, case, repetition, and calls to other modules. Sample flowcharts for several of the book's example programs are presented.

Features of the Text

Concept Statements	Each major section of the text starts with a concept statement. This statement summarizes the key idea of the section.
Example Programs	The text has over 350 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.
Tying It All Together	This special section, found at the end of most chapters, shows the student how to do something clever and fun with the material covered in that chapter.

VideoNotes

A series of online videos developed for this book are available for viewing at `http://www.pearson.com/gaddis`. VideoNote icons appear throughout the text, alerting the student to videos about specific topics.

Checkpoints

Checkpoints are questions placed throughout each chapter as a selftest study aid. Answers for all Checkpoint questions are provided in Appendix C at the back of the book so students can 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 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, with complete code provided for each one. Additional case studies are provided on the book's companion website. 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.

Each chapter ends with a *Soft Skills* exercise that focuses on communication and group process skills. Answers to the odd numbered review questions and review exercises are provided in Appendix D at the back of the book.

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.

Group Projects

There are a number of 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 teamwork within the classroom.

C++ Quick Reference Guide

For easy access, a quick reference guide to the C++ language is printed on the inside back cover.

Supplements

Student Resources

The following items are available on the Gaddis Series resource page at www.pearson.com/gaddis:

- Complete source code for every program included in the book
- Additional case studies, complete with source code
- A full set of appendices (including several tutorials) that accompany the book
- Access to the book's companion VideoNotes
- Links to download numerous programming environments and IDEs, including Visual Studio Community Edition.

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
- A computerized test bank
- A collection of lab exercises that accompany the introductory material
- Source code files

Visit the Pearson Education Instructor Resource Center (http://www.pearson.com) for information on how to access these.

Practice and Assessment with MyLab Programming

MyLab Programming helps students fully grasp the logic, semantics, and syntax of programming. Through practice exercises and immediate personalized feedback, *MyLab Programming* 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, *MyLab Programming* consists of hundreds of small practice exercises 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 help them figure out what went wrong. For instructors, a comprehensive gradebook tracks correct and incorrect answers and stores the code input by students for review.

MyLab Programming 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 *MyLab Programming* in your course, visit www.pearson.com/mylab/programming.

Which Gaddis C++ book is right for you?

The *Starting Out with C++* Series includes three books. One is sure to fit your course:

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