

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.

Copyright © 2020, 2017, 2014 Pearson Education, Inc., 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 likewise. For information regarding permissions, request forms and the appropriate contacts within the Pearson Education Global Rights & Permissions department, please visit www.pearson.com/permissions/.

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® Windows®, and Microsoft Office® 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.

The programs and applications presented in this book have been included for their instructional value. They have been tested with care, but are not guaranteed for any particular purpose. The publisher does not offer any warranties or representations, nor does it accept any liabilities with respect to the programs or applications.

Many of the designations by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed in initial caps or all caps.

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



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

Contents at a Glance

Preface xv

Chapter 1	Introduction to Computers and Programming	1
Chapter 2	Introduction to C++	29
Chapter 3	Expressions and Interactivity	79
Chapter 4	Making Decisions	157
Chapter 5	Looping	247
Chapter 6	Functions	327
Chapter 7	Introduction to Classes and Objects	411
Chapter 8	Arrays and Vectors	513
Chapter 9	Searching, Sorting, and Algorithm Analysis	613
Chapter 10	Pointers	659
Chapter 11	More about Classes and Object-Oriented Programming	717
Chapter 12	More on C-Strings and the <code>string</code> Class	821
Chapter 13	Advanced File and I/O Operations	867
Chapter 14	Recursion	929
Chapter 15	Polymorphism and Virtual Functions	963
Chapter 16	Exceptions and Templates	1001
Chapter 17	The Standard Template Library	1035
Chapter 18	Linked Lists	1131
Chapter 19	Stacks and Queues	1183
Chapter 20	Binary Trees	1223
	Appendix A: The ASCII Character Set	1253
	Appendix B: Operator Precedence and Associativity	1257
	Appendix C: Answers to Checkpoints	1259
	Appendix D: Answers to Odd-Numbered Review Questions	1301
	Index	1323

Contents

Preface xv

CHAPTER 1 Introduction to Computers and Programming 1

- 1.1 Why Program? 1
- 1.2 Computer Systems: Hardware and Software 3
- 1.3 Programs and Programming Languages 8
- 1.4 What Is a Program Made of? 13
- 1.5 Input, Processing, and Output 17
- 1.6 The Programming Process 18
- 1.7 Tying It All Together: *Hi! It's Me* 23

CHAPTER 2 Introduction to C++ 29

- 2.1 The Parts of a C++ Program 29
- 2.2 The `cout` Object 33
- 2.3 The `#include` Directive 38
- 2.4 Variables and the Assignment Statement 39
- 2.5 Literals 41
- 2.6 Identifiers 43
- 2.7 Integer Data Types 45
- 2.8 Floating-Point Data Types 50
- 2.9 The `char` Data Type 54
- 2.10 The C++ `string` Class 58
- 2.11 The `bool` Data Type 59
- 2.12 Determining the Size of a Data Type 60
- 2.13 More on Variable Assignments and Initialization 61
- 2.14 Scope 63
- 2.15 Arithmetic Operators 64
- 2.16 Comments 68
- 2.17 Programming Style 69
- 2.18 Tying It All Together: *Smile!* 71

CHAPTER 3 Expressions and Interactivity 79

- 3.1 The `cin` Object 79
- 3.2 Mathematical Expressions 86
- 3.3 Data Type Conversion and Type Casting 94
- 3.4 Overflow and Underflow 100
- 3.5 Named Constants 101
- 3.6 Multiple and Combined Assignment 104
- 3.7 Formatting Output 108
- 3.8 Working with Characters and Strings 118
- 3.9 More Mathematical Library Functions 132
- 3.10 Random Numbers 134
- 3.11 Focus on Debugging: *Hand Tracing a Program* 138
- 3.12 Green Fields Landscaping Case Study—Part 1 140
- 3.13 Tying It All Together: *Word Game* 143

CHAPTER 4 Making Decisions 157

- 4.1 Relational Operators 157
- 4.2 The `if` Statement 163
- 4.3 The `if/else` Statement 172
- 4.4 The `if/else if` Statement 177
- 4.5 Menu-Driven Programs 185
- 4.6 Nested `if` Statements 187
- 4.7 Logical Operators 191
- 4.8 Validating User Input 200
- 4.9 More about Blocks and Scope 202
- 4.10 More about Characters and Strings 205
- 4.11 The Conditional Operator 211
- 4.12 The `switch` Statement 215
- 4.13 Enumerated Data Types 224
- 4.14 Focus on Testing and Debugging: *Validating Output Results* 227
- 4.15 Green Fields Landscaping Case Study—Part 2 229
- 4.16 Tying It All Together: *Fortune Teller* 234

CHAPTER 5 Looping 247

- 5.1 Introduction to Loops: The `while` Loop 247
- 5.2 Using the `while` Loop for Input Validation 254
- 5.3 The Increment and Decrement Operators 257
- 5.4 Counters 262
- 5.5 Keeping a Running Total 264
- 5.6 Sentinels 267
- 5.7 The `do-while` Loop 269

- 5.8 The for Loop 275
- 5.9 Deciding Which Loop to Use 281
- 5.10 Nested Loops 283
- 5.11 Breaking Out of a Loop 285
- 5.12 Using Files for Data Storage 288
- 5.13 Focus on Testing and Debugging: *Creating Good Test Data* 304
- 5.14 Central Mountain Credit Union Case Study 307
- 5.15 Tying It All Together: *What a Colorful World* 311

CHAPTER 6 Functions 327

- 6.1 Modular Programming 327
- 6.2 Defining and Calling Functions 328
- 6.3 Function Prototypes 336
- 6.4 Sending Data into a Function 337
- 6.5 Passing Data by Value 342
- 6.6 The return Statement 346
- 6.7 Returning a Value from a Function 347
- 6.8 Returning a Boolean Value 353
- 6.9 Using Functions in a Menu-Driven Program 355
- 6.10 Local and Global Variables 359
- 6.11 Static Local Variables 366
- 6.12 Default Arguments 368
- 6.13 Using Reference Variables as Parameters 372
- 6.14 Overloading Functions 382
- 6.15 The `exit()` Function 386
- 6.16 Stubs and Drivers 389
- 6.17 Little Lotto Case Study 391
- 6.18 Tying It All Together: *Glowing Jack-o-lantern* 396

CHAPTER 7 Introduction to Classes and Objects 411

- 7.1 Abstract Data Types 411
- 7.2 Object-Oriented Programming 413
- 7.3 Introduction to Classes 415
- 7.4 Creating and Using Objects 418
- 7.5 Defining Member Functions 420
- 7.6 Constructors 427
- 7.7 Destructors 435
- 7.8 Private Member Functions 438
- 7.9 Passing Objects to Functions 441
- 7.10 Object Composition 448
- 7.11 Separating Class Specification, Implementation, and Client Code 452

- 7.12 Structures 459
- 7.13 More about Enumerated Data Types 471
- 7.14 Home Software Company OOP Case Study 475
- 7.15 Introduction to Object-Oriented Analysis and Design 482
- 7.16 Screen Control 492
- 7.17 Tying It All Together: *Yoyo Animation* 497

CHAPTER 8 Arrays and Vectors 513

- 8.1 Arrays Hold Multiple Values 513
- 8.2 Accessing Array Elements 515
- 8.3 Inputting and Displaying Array Data 517
- 8.4 Array Initialization 524
- 8.5 The Range-Based for Loop 531
- 8.6 Processing Array Contents 534
- 8.7 Using Parallel Arrays 546
- 8.8 The typedef Statement 550
- 8.9 Arrays as Function Arguments 550
- 8.10 Two-Dimensional Arrays 560
- 8.11 Arrays with Three or More Dimensions 567
- 8.12 Introduction to the STL vector 571
- 8.13 Arrays of Objects 583
- 8.14 National Commerce Bank Case Study 593
- 8.15 Tying It All Together: *Rock, Paper, Scissors* 595

CHAPTER 9 Searching, Sorting, and Algorithm Analysis 613

- 9.1 Introduction to Search Algorithms 613
- 9.2 Searching an Array of Objects 620
- 9.3 Introduction to Sorting Algorithms 623
- 9.4 Sorting an Array of Objects 634
- 9.5 Sorting and Searching Vectors 637
- 9.6 Introduction to Analysis of Algorithms 641
- 9.7 Case Studies 649
- 9.8 Tying It All Together: *Secret Messages* 649

CHAPTER 10 Pointers 659

- 10.1 Pointers and the Address Operator 659
- 10.2 Pointer Variables 661
- 10.3 The Relationship Between Arrays and Pointers 665

- 10.4 Pointer Arithmetic 669
- 10.5 Initializing Pointers 670
- 10.6 Comparing Pointers 673
- 10.7 Pointers as Function Parameters 675
- 10.8 Pointers to Constants and Constant Pointers 679
- 10.9 Dynamic Memory Allocation 684
- 10.10 Returning Pointers from Functions 688
- 10.11 Pointers to Class Objects and Structures 694
- 10.12 Selecting Members of Objects 698
- 10.13 Smart Pointers 700
- 10.14 Tying It All Together: *Pardon Me, Do You Have the Time?* 708

CHAPTER 11 More about Classes and Object-Oriented Programming 717

- 11.1 The `this` Pointer and Constant Member Functions 717
- 11.2 Static Members 722
- 11.3 Friends of Classes 729
- 11.4 Memberwise Assignment 734
- 11.5 Copy Constructors 735
- 11.6 Operator Overloading 744
- 11.7 Rvalue References and Move Operations 765
- 11.8 Type Conversion Operators 775
- 11.9 Convert Constructors 778
- 11.10 Aggregation and Composition 782
- 11.11 Inheritance 787
- 11.12 Protected Members and Class Access 792
- 11.13 Constructors, Destructors, and Inheritance 798
- 11.14 Overriding Base Class Functions 803
- 11.15 Tying It All Together: *Putting Data on the World Wide Web* 806

CHAPTER 12 More on C-Strings and the `string` Class 821

- 12.1 Character Testing 821
- 12.2 Character Case Conversion 825
- 12.3 C-Strings 828
- 12.4 Library Functions for Working with C-Strings 833
- 12.5 Conversions Between Numbers and Strings 842
- 12.6 Writing Your Own C-String Handling Functions 846
- 12.7 More about the C++ `string` Class 852
- 12.8 Advanced Software Enterprises Case Study 855
- 12.9 Tying It All Together: *Program Execution Environments* 857

CHAPTER 13 Advanced File and I/O Operations 867

- 13.1 Input and Output Streams 867
- 13.2 More Detailed Error Testing 875
- 13.3 Member Functions for Reading and Writing Files 879
- 13.4 Binary Files 891
- 13.5 Creating Records with Structures 895
- 13.6 Random-Access Files 900
- 13.7 Opening a File for Both Input and Output 907
- 13.8 Online Friendship Connections Case Study 912
- 13.9 Tying It All Together: *File Merging and Color-Coded HTML* 917

CHAPTER 14 Recursion 929

- 14.1 Introduction to Recursion 929
- 14.2 The Recursive Factorial Function 936
- 14.3 The Recursive gcd Function 938
- 14.4 Solving Recursively Defined Problems 939
- 14.5 A Recursive Binary Search Function 941
- 14.6 The QuickSort Algorithm 943
- 14.7 The Towers of Hanoi 947
- 14.8 Exhaustive and Enumeration Algorithms 950
- 14.9 Recursion versus Iteration 954
- 14.10 Tying It All Together: *Infix and Prefix Expressions* 955

CHAPTER 15 Polymorphism and Virtual Functions 963

- 15.1 Type Compatibility in Inheritance Hierarchies 963
- 15.2 Polymorphism and Virtual Member Functions 969
- 15.3 Abstract Base Classes and Pure Virtual Functions 977
- 15.4 Composition versus Inheritance 983
- 15.5 Secure Encryption Systems, Inc., Case Study 987
- 15.6 Tying It All Together: *Let's Move It* 990

CHAPTER 16 Exceptions and Templates 1001

- 16.1 Exceptions 1001
- 16.2 Function Templates 1013
- 16.3 Class Templates 1021
- 16.4 Class Templates and Inheritance 1026

Chapter 17 The Standard Template Library 1035

- 17.1 Introduction to the Standard Template Library 1035
- 17.2 STL Container and Iterator Fundamentals 1035
- 17.3 The vector Class 1047

- 17.4 The `map`, `multimap`, and `unordered_map` Classes 1060
- 17.5 The `set`, `multiset`, and `unordered_set` Classes 1086
- 17.6 Algorithms 1093
- 17.7 Introduction to Function Objects and Lambda Expressions 1111
- 17.8 Tying It All Together: *Word Transformers Game* 1118

CHAPTER 18 Linked Lists 1131

- 18.1 Introduction to Linked Lists 1131
- 18.2 Linked List Operations 1137
- 18.3 A Linked List Template 1149
- 18.4 Recursive Linked List Operations 1153
- 18.5 Variations of the Linked List 1161
- 18.6 The STL `list` and `forward_list` Containers 1163
- 18.7 Reliable Software Systems, Inc., Case Study 1167
- 18.8 Tying It All Together: *More on Graphics and Animation* 1170

CHAPTER 19 Stacks and Queues 1183

- 19.1 Introduction to Stacks 1183
- 19.2 Dynamic Stacks 1191
- 19.3 The STL `stack` Container 1195
- 19.4 Introduction to Queues 1197
- 19.5 Dynamic Queues 1204
- 19.6 The STL `deque` and `queue` Containers 1207
- 19.7 Eliminating Recursion 1210
- 19.8 Tying It All Together: *Converting Postfix Expressions to Infix* 1215

CHAPTER 20 Binary Trees 1223

- 20.1 Definition and Applications of Binary Trees 1223
- 20.2 Binary Search Tree Operations 1227
- 20.3 Template Considerations for Binary Search Trees 1243
- 20.4 Tying It All Together: *Genealogy Trees* 1243

Appendix A The ASCII Character Set 1253

Appendix B Operator Precedence and Associativity 1257

Appendix C Answers to Checkpoints 1259

Appendix D Answers to Odd-Numbered Review Questions 1301

Index 1323

Additional Appendices

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

Appendix E: A Brief Introduction to Object-Oriented Programming

Appendix F: Using UML in Class Design

Appendix G: Multi-Source File Programs

Appendix H: Multiple and Virtual Inheritance

Appendix I: Header File and Library Function Reference

Appendix J: Namespaces

Appendix K: C++ Casts and Run-Time Type Identification

Appendix L: Passing Command Line Arguments

Appendix M: Binary Numbers and Bitwise Operations

Appendix N: Introduction to Flowcharting

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)*



Chapter 10	Pointer Variables, p. 661 Dynamically Allocating an Array, p. 685 Solving the Days in Current Month Problem, p. 716
Chapter 11	Operator Overloading, p. 744 Aggregation and Composition, p. 782 Overriding Base Class Functions, p. 803 Solving the Number of Days Worked Problem, p. 817
Chapter 12	Converting Strings to Numbers, p. 842 Writing a C-String Handling Function, p. 846 Solving the Case Manipulator Problem, p. 863
Chapter 13	The get Family of Member Functions, p. 883 Rewinding a File, p. 887 Solving the File Encryption Filter Problem, p. 926
Chapter 14	Recursive Binary Search, p. 941 QuickSort, p. 943 Solving the Recursive Multiplication Problem, p. 961
Chapter 15	Polymorphism, p. 969 Composition versus Inheritance, p. 983 Solving the Sequence Sum Problem, p. 999
Chapter 16	Throwing and Handling Exceptions, p. 1002 Writing a Function Template, p. 1014 Solving the Arithmetic Exceptions Problem, p. 1032
Chapter 17	The array Container, p. 1038 Iterators, p. 1040 The vector Container, p. 1047 The map Container, p. 1061 The set Container, p. 1086 Function Objects and Lambda Expressions, p. 1111 The Course Information Problem, p. 1126
Chapter 18	Adding an Element to a Linked List, p. 1139 Removing an Element from a Linked List, p. 1146 Solving the Member Insertion by Position Problem, p. 1180
Chapter 19	Storing Objects in an STL Stack, p. 1195 Storing Objects in an STL Queue, p. 1209 Solving the File Reverser Problem, p. 1221
Chapter 20	Inserting an Element into a Binary Tree, p. 1230 Removing an Element from a Binary Tree, p. 1234 Solving the Tree Size Problem, p. 1250

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.

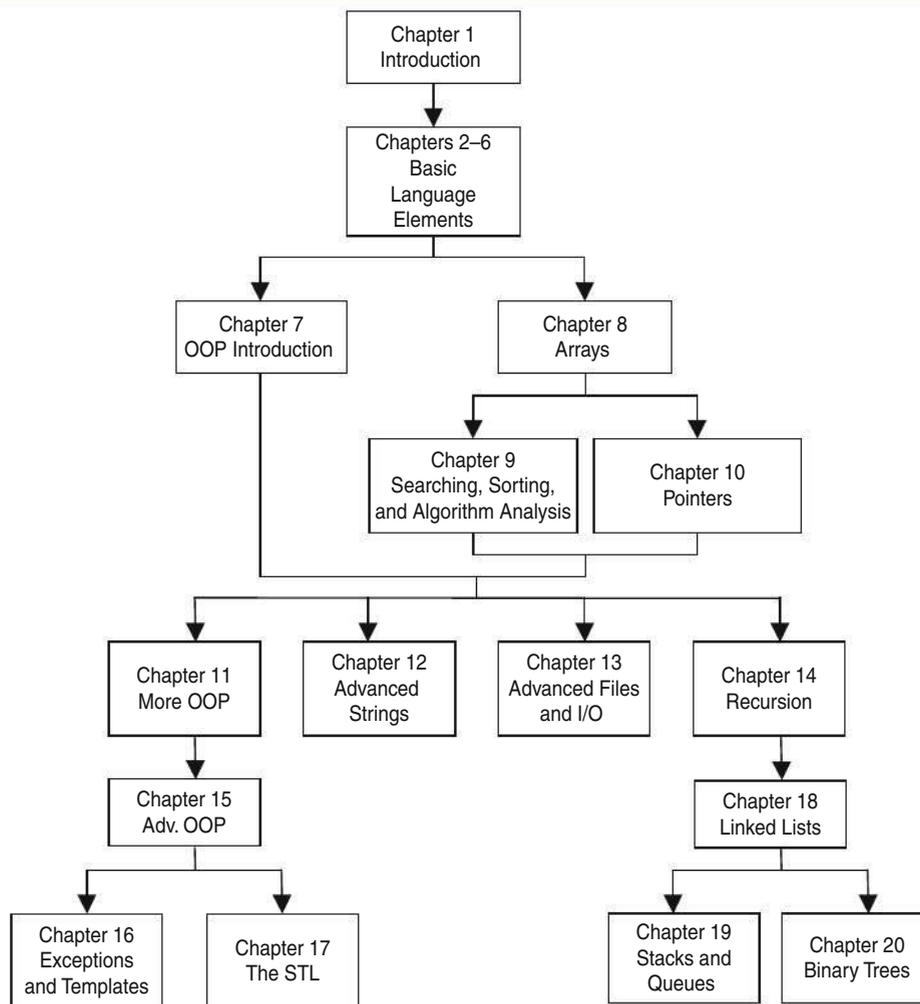
Instructors who prefer to introduce arrays before classes can cover Chapter 8 before Chapter 7. In this case it is only necessary to postpone Section 8.13 (Arrays of Objects) until Chapter 7 has been covered.

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



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_ptr`s and `weak_ptr`s, 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

- | | |
|------------------------------|--|
| <i>Concept Statements</i> | Each major section of the text starts with a concept statement. This statement summarizes the key idea of the section. |
| <i>Example Programs</i> | 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. |
| <i>Program Output</i> | After each example program there is a sample of its screen output. This immediately shows the student how the program should function. |
| <i>Tying It All Together</i> | 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***

VideoNote

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 self-test 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*