

ABSOLUTE C++

WALTER SAVITCH

University of California, San Diego



Boston San Francisco New York London
Toronto Sydney Tokyo Singapore Madrid Mexico City
Munich Paris Cape Town Hong Kong Montreal

Executive Editor	Susan Hartman Sullivan
Executive Marketing Manager	Michael Hirsch
Project Management	Argosy Publishing
Composition and Art	Argosy Publishing
Copyeditor	Cindy Kogut
Proofreader	Bob LaRoche
Indexer	Larry Sweazy
Text and Cover Design	Leslie Haimes
Cover Photo	Renee Lynn/Stone by Getty Images
Design Manager	Gina Hagen Kolenda
Prepress and Manufacturing	Caroline Fell

Access the latest information about Addison-Wesley titles from our World Wide Web site:
<http://www.aw.com/cs>

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

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.

Library of Congress Cataloging-in-Publication Data

Savitch, Walter J., 1943-

Absolute C++ / Walter Savitch

p. cm.

ISBN 0-201-70927-9 (pbk.)

1. C++ (Computer program language) I. Title.

QA76.73.C153 S279 2002

005.13'3--dc212001046383

Copyright © 2002 by Pearson Education, Inc.

Chapter Opener Image: © 2002 PhotoDisc, Inc.

All rights reserved. No part of this publication may be reproduced, stored in a retrieval system, or transmitted, in any form or by any means, electronic, mechanical, photocopying, recording, or otherwise, without the prior written permission of the publisher. Printed in the United States of America.

ISBN 0-201-70927-9

12345678910-DOC-04030201

Preface

This book is designed to be a text book and reference for programming in the C++ language. Although it does include programming techniques, it is organized around the features of the C++ language, rather than any particular curriculum of programming techniques. The main audience I had in mind when writing this book were undergraduate students who have had some programming experience but who had not yet learned the C++ language. As such it would be a suitable C++ text or reference for a second or later computer science course that uses C++ and could even be used for a first programming course. This book is designed to accommodate a wide range of users. The beginning chapters are written at a level that is accessible to beginners, while the boxed sections of those chapters serve to quickly introduce more experienced programmers to basic C++ syntax. Later chapters are still designed to be accessible, but are written at a level suitable for students who have progressed to these more advanced topics. (For those who want a beginning textbook with more pedagogical material and more on very basic programming technique, I suggest another book I wrote that is more along these lines.)

This book is also suitable for somebody learning the C++ language on their own. It is accessible for selfstudy and complete as a reference.

The C++ coverage in this book is very complete. It goes well beyond what a beginner needs to know. In particular it has extensive coverage of inheritance, polymorphism, and exception handling in C++. It also has an extensive introduction to the Standard Template Library (STL).

This book also includes an introduction to Patterns and UML. Since many computer science curricula postpone recursions to a second computer science course, it includes a full chapter on recursion.

More on the versatility of the text and other special features are given in the remainder of this preface.

ANSI/ISO C++ STANDARD

This book was written to conform to the new ANSI/ISO C++ Standard.

STANDARD TEMPLATE LIBRARY

The Standard Template Library is an extensive library collection of preprogrammed data structure classes and important algorithms. The STL is perhaps as big a topic as the core C++ language. However, this book does contain a very substantial introduction to the STL. There is a full chapter on the general topic of templates, a full chapter on the particulars of the STL, as well as other material on or related to the STL at other points in the text.

OBJECT-ORIENTED PROGRAMMING

This book is organized around the structure of the C++ language. As such, the earlier chapters which cover aspects of C++ that are common to most all high level programming languages are not particularly oriented toward OOP programming. As a reference book and as a book for learning a second language this makes sense. However, I do consider C++ to be an OOP language. If you are really programming in C++ and not C, then you must be availing yourself of the OOP features of C++. This book gives extensive coverage of encapsulation, inheritance, and polymorphism as realized in the C++ language. The final Chapter on Patterns and UML gives additional coverage of OOP related material.

FLEXIBILITY IN TOPIC ORDERING

This book allows instructors wide latitude in reordering the material. This is important if a book is to serve as a reference. This is also in keeping with my philosophy of writing books that accommodate themselves to an instructor's style, rather than tying the instructor to an author's personal preference of topic ordering. With this in mind, each chapter introduction explains what material must be covered before doing each section of the chapter.

ACCESSIBLE TO STUDENTS

It is not enough for a book to present the right topics in the right order. It is not even enough for it be clear and correct when read by an instructor or other expert. The material needs to be presented in a way that is accessible to the person who does not yet

know the material. Like my other text books that proved to be very popular with students, this book was written to be friendly and accessible to the student.

SUMMARY BOXES

Each major point is summarized in a boxed section. These boxed sections are spread throughout each chapter. They serve as summaries of the material, as a quick reference source, and as way to quickly learn the C++ syntax for feature you know about in general but for which you do not know the C++ particulars.

SELF-TEST EXERCISES

Each chapter contains numerous Self-Test Exercises at strategic points in the chapter. Complete answers for all the Self-Test Exercises are given at the end of each chapter.

OTHER FEATURES

Pitfall sections, programming technique sections, and examples of complete programs with sample I/O are given throughout each chapter. Each chapter ends with a summary section and a collection of programming projects suitable to assign to students.

SUPPORT MATERIAL

A full package of support material is available to instructors who adopt this book. They include: an instructor's manual, all the source code from the text, PowerPoint slides, a computerized testbank, and a free copy of the Microsoft Visual C++ 6.0 Introductory edition.

The source code and other information is available to all readers at the book's companion website.

ACKNOWLEDGMENTS

Numerous individuals have contributed invaluable help and support in making this book happen. Frank Ruggirello and my editor Susan Hartman at Addison-Wesley are the ones who first conceived of the idea for this book. Susan Hartman, Galia Shokry, Lisa Kalner, and the other fine people at Addison-Wesley were a continuing source of support and encouragement in getting the book reviewed, revised, and out the door.

Cindy Kogut did an incredible thorough job of copy editing. Sally Boylan and others at Argosy Publishing did great work under rushed conditions in converting the manuscript to type set pages.

David Teague deserves special acknowledgment. I very much appreciate his hard work, good insights, and careful researching for this book.

I thank my good friend Mario Lopez for the many helpful conversations we had about C++.

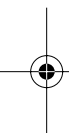
The following list of reviewers provided corrections and suggestions that contributed greatly to the final product. I thank them all. In random order they are: Kenrick Mock, University of Alaska, Anchorage; Richard Albright, University of Delaware; H.E. Dunsmore, Purdue University; Christopher E. Cramer; Drue Coles, Boston University; Evan Golub, University of Maryland; Stephen Corbesero, Moravian College; Fredrick H. Colclough, Colorado Technical University; Joel Weinstein, Northeastern University; Stephen P. Leach, Florida State University; Alvin S. Lim, Auburn University; Martin Dulberg, North Carolina State University.

I again thank David Teague. This time for his excellent work in preparing the instructor's guide.

Finally, I thank Christina for putting up with my working late on the book and even offering encouragement instead of complaining.

W.S.

<http://www-cse.ucsd.edu/users/savitch/>
wsavitch@ucsd.edu



Contents

Chapter 1 C++ Basics 1

1.1 INTRODUCTION TO C++ 2

- Origins of the C++ Language 2
- C++ and Object-Oriented Programming 3
- The Character of C++ 3
- C++ Terminology 4
- A Sample C++ Program 4

1.2 VARIABLES, EXPRESSIONS, AND ASSIGNMENT STATEMENTS 6

- Identifiers 6
- Variables 8
- Assignment Statements 10
 - PITFALL** Uninitialized Variables 12
 - TIP** Use Meaningful Names 13
- More Assignment Statements 13
- Assignment Compatibility 14
- Literals 15
- Escape Sequences 17
- Naming Constants 17
- Arithmetic Operators and Expressions 19
- Integer and Floating-Point Division 21
 - PITFALL** Division with Whole Numbers 22
- Type Casting 23
- Increment and Decrement Operators 25
 - PITFALL** Order of Evaluation 27

1.3 CONSOLE INPUT/OUTPUT 28

- Output Using `cout` 28
- New Lines in Output 29
 - TIP** End Each Program with `\n` or `endl` 30
- Formatting for Numbers with a Decimal Point 30
- Output with `cerr` 32
- Input Using `cin` 32
 - TIP** Line Breaks in I/O 34

1.4 PROGRAM STYLE 35

- Comments 35

1.5 LIBRARIES AND NAMESPACES 36

Libraries and include Directives 36

Namespaces 37

PITFALL Problems with Library Names 38**CHAPTER SUMMARY 38****ANSWERS TO SELF-TEST EXERCISES 39****PROGRAMMING PROJECTS 41****Chapter 2 Flow of Control 43****2.1 BOOLEAN EXPRESSIONS 44**

Building Boolean Expressions 44

PITFALL Strings of Inequalities 45

Evaluating Boolean Expressions 46

Precedence Rules 48

PITFALL Integer Values Can Be Used as Boolean Values 52**2.2 BRANCHING MECHANISMS 54**

if-else Statements 54

Compound Statements 56

PITFALL Using = in Place of == 57

Omitting the else 58

Nested Statements 59

Multiway if-else Statement 59

The switch Statement 61

PITFALL Forgetting a break in a switch Statement 63**TIP** Use switch Statements for Menus 63

Enumeration Types 64

The Conditional Operator 64

2.3 LOOPS 66

The while and do-while Statements 66

Increment and Decrement Operators Revisited 69

The Comma Operator 72

The for Statement 73

TIP Repeat-N-Times Loops 76**PITFALL** Extra Semicolon in a for Statement 76**PITFALL** Infinite Loops 77

The break and continue Statements 80

Nested Loops 83

CHAPTER SUMMARY	83
ANSWERS TO SELF-TEST EXERCISES	84
PROGRAMMING PROJECTS	89

Chapter 3 Function Basics 91

3.1 PREDEFINED FUNCTIONS	92
Predefined Functions That Return a Value	92
Predefined void Functions	97
A Random Number Generator	99
3.2 PROGRAMMER-DEFINED FUNCTIONS	103
Defining Functions That Return a Value	103
Alternate Form for Function Declarations	106
PITFALL Arguments in the Wrong Order	107
PITFALL Use of the Terms Parameter and Argument	107
Functions Calling Functions	107
EXAMPLE A Rounding Function	107
Functions That Return a Boolean Value	110
Defining void Functions	111
return Statements in void Functions	112
Preconditions and Postconditions	113
main Is a Function	115
Recursive Functions	116
3.3 SCOPE RULES	117
Local Variables	117
Procedural Abstraction	120
Global Constants and Global Variables	121
Blocks	124
Nested Scopes	124
TIP Use Function Calls in Branching and Loop Statements	125
Variables Declared in a for Loop	125

CHAPTER SUMMARY	126
ANSWERS TO SELF-TEST EXERCISES	127
PROGRAMMING PROJECTS	130

Chapter 4 Parameters and Overloading 133

4.1 PARAMETERS	134
Call-by-Value Parameters	134
A First Look at Call-by-Reference Parameters	137

x Contents

Call-by-Reference Mechanism in Detail 139

EXAMPLE The swapValues Function 141

Constant Reference Parameters 142

TIP Think of Actions, Not Code 143

Mixed Parameter Lists 144

TIP What Kind of Parameter to Use 145

PITFALL Inadvertent Local Variables 146

TIP Choosing Formal Parameter Names 147

EXAMPLE Buying Pizza 147

4.2 OVERLOADING AND DEFAULT ARGUMENTS 151

Introduction to Overloading 151

PITFALL Automatic Type Conversion and Overloading 154

Rules for Resolving Overloading 156

EXAMPLE Revised Pizza-Buying Program 157

Default Arguments 159

4.3 TESTING AND DEBUGGING FUNCTIONS 161

The assert Macro 161

Stubs and Drivers 162

CHAPTER SUMMARY 165

ANSWERS TO SELF-TEST EXERCISES 166

PROGRAMMING PROJECTS 168

Chapter 5 Arrays 171

5.1 INTRODUCTION TO ARRAYS 172

Declaring and Referencing Arrays 172

TIP Use for Loops with Arrays 175

PITFALL Array Indexes Always Start with Zero 175

TIP Use a Defined Constant for the Size of an Array 175

Arrays in Memory 176

PITFALL Array Index Out of Range 177

Initializing Arrays 178

5.2 ARRAYS IN FUNCTIONS 181

Indexed Variables as Function Arguments 181

Entire Arrays as Function Arguments 182

The const Parameter Modifier 185

PITFALL Inconsistent Use of const Parameters 187

Functions That Return an Array 188

EXAMPLE Production Graph 188

5.3	PROGRAMMING WITH ARRAYS	194
	Partially Filled Arrays	194
	TIP Do Not Skimp on Formal Parameters	194
	EXAMPLE Searching an Array	197
	EXAMPLE Sorting an Array	199
5.4	MULTIDIMENSIONAL ARRAYS	204
	Multidimensional Array Basics	204
	Multidimensional Array Parameters	205
	EXAMPLE Two-Dimensional Grading Program	207
	CHAPTER SUMMARY	211
	ANSWERS TO SELF-TEST EXERCISES	212
	PROGRAMMING PROJECTS	216

Chapter 6 Structures and Classes 223

6.1	STRUCTURES	224
	Structure Types	226
	PITFALL Forgetting a Semicolon in a Structure Definition	230
	Structures as Function Arguments	230
	TIP Use Hierarchical Structures	231
	Initializing Structures	234
6.2	CLASSES	236
	Defining Classes and Member Functions	236
	Encapsulation	242
	Public and Private Members	243
	Accessor and Mutator Functions	247
	TIP Separate Interface and Implementation	248
	TIP A Test for Encapsulation	249
	Structures versus Classes	250
	TIP Thinking Objects	252
	CHAPTER SUMMARY	252
	ANSWERS TO SELF-TEST EXERCISES	253
	PROGRAMMING PROJECTS	255

Chapter 7 Constructors and Other Tools 257

7.1	CONSTRUCTORS	258
	Constructor Definitions	258
	PITFALL Constructors with No Arguments	263
	Explicit Constructor Calls	265

TIP Always Include a Default Constructor 265

EXAMPLE BankAccount Class 268

Class Type Member Variables 274

7.2 MORE TOOLS 277

The const Parameter Modifier 277

PITFALL Inconsistent Use of const 279

Inline Functions 284

Static Members 286

Nested and Local Class Definitions 289

7.3 VECTORS—A PREVIEW OF THE STANDARD TEMPLATE LIBRARY 290

Vector Basics 290

PITFALL Using Square Brackets beyond the Vector Size 293

TIP Vector Assignment Is Well Behaved 294

Efficiency Issues 294

CHAPTER SUMMARY 296

ANSWERS TO SELF-TEST EXERCISES 296

PROGRAMMING PROJECTS 298

Chapter 8 Operator Overloading, Friends, and References 301

8.1 BASIC OPERATOR OVERLOADING 302

Overloading Basics 303

TIP A Constructor Can Return an Object 308

Returning by const Value 309

TIP Returning Member Variables of a Class Type 312

Overloading Unary Operators 313

Overloading as Member Functions 314

TIP A Class Has Access to All Its Objects 316

Overloading Function Application () 317

PITFALL Overloading &&, ||, and the Comma Operator 317

8.2 FRIEND FUNCTIONS AND AUTOMATIC TYPE CONVERSION 318

Constructors for Automatic Type Conversion 318

PITFALL Member Operators and Automatic Type Conversion 319

Friend Functions 320

PITFALL Compilers without Friends 323

Friend Classes 323

8.3 REFERENCES AND MORE OVERLOADED OPERATORS 324

References 325

PITFALL Returning a Reference to Certain Member Variables 327

Overloading >> and <<	327
TIP What Mode of Returned Value to Use	335
The Assignment Operator	336
Overloading the Increment and Decrement Operators	337
Overloading the Array Operator []	337
Overloading Based on L-Value versus R-Value	341
CHAPTER SUMMARY	342
ANSWERS TO SELF-TEST EXERCISES	342
PROGRAMMING PROJECTS	345

Chapter 9 Strings 349

9.1 AN ARRAY TYPE FOR STRINGS	351
C-String Values and C-String Variables	351
PITFALL Using = and == with C-strings	355
Other Functions in <cstring>	357
C-String Input and Output	360
9.2 CHARACTER MANIPULATION TOOLS	363
Character I/O	363
The Member Functions get and put	364
EXAMPLE Checking Input Using a Newline Function	366
PITFALL Unexpected '\n' in Input	368
The putback, peek, and ignore Member Functions	369
Character-Manipulating Functions	372
PITFALL toupper and tolower Return int Values	374
9.3 THE STANDARD CLASS string	375
Introduction to the Standard Class string	376
I/O with the Class string	379
TIP More Versions of getline	382
PITFALL Mixing cin >> variable; and getline	383
String Processing with the Class string	384
EXAMPLE Palindrome Testing	388
Converting between string Objects and C-Strings	392
CHAPTER SUMMARY	393
ANSWERS TO SELF-TEST EXERCISES	393
PROGRAMMING PROJECTS	397

Chapter 10 Pointers and Dynamic Arrays 403

10.1 POINTERS 404

Pointer Variables 405

Basic Memory Management 414

PITFALL Dangling Pointers 416

Dynamic Variables and Automatic Variables 416

TIP Define Pointer Types 417

PITFALL Pointers as Call-by-Value Parameters 419

Uses for Pointers 421

10.2 DYNAMIC ARRAYS 422

Array Variables and Pointer Variables 422

Creating and Using Dynamic Arrays 423

EXAMPLE A Function That Returns an Array 427

Pointer Arithmetic 429

Multidimensional Dynamic Arrays 430

10.3 CLASSES, POINTERS, AND DYNAMIC ARRAYS 433

The \rightarrow Operator 433

The `this` Pointer 434

Overloading the Assignment Operator 435

EXAMPLE A Class for Partially Filled Arrays 437

Destructors 445

Copy Constructors 446

CHAPTER SUMMARY 451

ANSWERS TO SELF-TEST EXERCISES 452

PROGRAMMING PROJECTS 454

Chapter 11 Separate Compilation and Namespaces 457

11.1 SEPARATE COMPILATION 458

Encapsulation Reviewed 459

Header Files and Implementation Files 460

EXAMPLE `DigitalTime` Class 468

TIP Reusable Components 469

Using `#ifndef` 469

TIP Defining Other Libraries 472

11.2 NAMESPACES 473

Namespaces and using Directives 473

Creating a Namespace 475

Using Declarations	478
Qualifying Names	480
EXAMPLE A Class Definition in a Namespace	482
TIP Choosing a Name for a Namespace	482
Unnamed Namespaces	484
PITFALL Confusing the Global Namespace and the Unnamed Namespace	490
TIP Unnamed Namespaces Replace the static Qualifier	491
TIP Hiding Helping Functions	491
Nested Namespaces	491
TIP What Namespace Specification Should You Use?	492
CHAPTER SUMMARY	495
ANSWERS TO SELF-TEST EXERCISES	495
PROGRAMMING PROJECTS	497

Chapter 12 Streams and File I/O 499

12.1	I/O STREAMS	501
	File I/O	501
	PITFALL Restrictions on Stream Variables	506
	Appending to a File	506
	TIP Another Syntax for Opening a File	508
	TIP Check That a File Was Opened Successfully	509
	Character I/O	512
	Checking for the End of a File	513
12.2	TOOLS FOR STREAM	I/O 517
	File Names as Input	517
	Formatting Output with Stream Functions	518
	Manipulators	521
	Saving Flag Settings	523
	More Output Stream Member Functions	524
	EXAMPLE Cleaning Up a File Format	525
	EXAMPLE Editing a Text File	528
12.3	STREAM HIERARCHIES: A PREVIEW OF INHERITANCE	528
	Inheritance among Stream Classes	528
	EXAMPLE Another newLine Function	533
12.4	Random Access to Files	536
	CHAPTER SUMMARY	538
	ANSWERS TO SELF-TEST EXERCISES	539
	PROGRAMMING PROJECTS	541

Chapter 13 Recursion 547

13.1 RECURSIVE VOID FUNCTIONS 549

EXAMPLE Vertical Numbers 549

Tracing a Recursive Call 552

A Closer Look at Recursion 555

PITFALL Infinite Recursion 556

Stacks for Recursion 558

PITFALL Stack Overflow 559

Recursion versus Iteration 559

13.2 RECURSIVE FUNCTIONS THAT RETURN A VALUE 561

General Form for a Recursive Function That Returns a Value 561

EXAMPLE Another Powers Function 561

13.3 THINKING RECURSIVELY 566

Recursive Design Techniques 566

Binary Search 568

CHAPTER SUMMARY 576

ANSWERS TO SELF-TEST EXERCISES 576

PROGRAMMING PROJECTS 581

Chapter 14 Inheritance 583

14.1 INHERITANCE BASICS 584

Derived Classes 584

Constructors in Derived Classes 594

PITFALL Use of Private Member Variables from the Base Class 596

PITFALL Private Member Functions Are Effectively Not Inherited 598

The protected Qualifier 598

Redefinition of Member Functions 601

Redefining versus Overloading 603

Access to a Redefined Base Function 604

Functions That Are Not Inherited 605

14.2 PROGRAMMING WITH INHERITANCE 606

Assignment Operators and Copy Constructors in Derived Classes 606

Destructors in Derived Classes 607

EXAMPLE Partially Filled Array with Backup 608

PITFALL Same Object on Both Sides of the Assignment Operator 617

EXAMPLE Alternate Implementation of PFArrayDBak 617

TIP A Class Has Access to Private Members of All Objects of the Class 618

TIP "Is a" versus "Has a" 620
Protected and Private Inheritance 621
Multiple Inheritance 622

CHAPTER SUMMARY 623
ANSWERS TO SELF-TEST EXERCISES 623
PROGRAMMING PROJECTS 625

Chapter 15 Polymorphism and Virtual Functions 627

15.1 VIRTUAL FUNCTION BASICS 628

Late Binding 628
Virtual Functions in C++ 629
TIP The Virtual Property Is Inherited 636
TIP When to Use a Virtual Function 636
PITFALL Omitting the Definition of a Virtual Member Function 637
Abstract Classes and Pure Virtual Functions 637
EXAMPLE An Abstract Class 638

15.2 POINTERS AND VIRTUAL FUNCTIONS 641

Virtual Functions and Extended Type Compatibility 641
PITFALL The Slicing Problem 645
TIP Make Destructors Virtual 646
Downcasting and Upcasting 647
How C++ Implements Virtual Functions 649

CHAPTER SUMMARY 650
ANSWERS TO SELF-TEST EXERCISES 651
PROGRAMMING PROJECTS 651

Chapter 16 Templates 653

16.1 FUNCTION TEMPLATES 654

Syntax for Function Templates 656
PITFALL Compiler Complications 659
EXAMPLE A Generic Sorting Function 661
TIP How to Define Templates 665
PITFALL Using a Template with an Inappropriate Type 665

16.2 CLASS TEMPLATES 667

Syntax for Class Templates 667
EXAMPLE An Array Template Class 671
The vector and basic_string Templates 677

16.3 TEMPLATES AND INHERITANCE 678**EXAMPLE** Template Class for a Partially Filled Array with Backup 678**CHAPTER SUMMARY** 684**ANSWERS TO SELF-TEST EXERCISES** 684**PROGRAMMING PROJECTS** 688**Chapter 17 Linked Data Structures 689****17.1 NODES AND LINKED LISTS 691**

Nodes 691

Linked Lists 696

Inserting a Node at the Head of a List 698

PITFALL Losing Nodes 700

Inserting and Removing Nodes Inside a List 702

PITFALL Using the Assignment Operator with Dynamic Data Structures 706

Searching a Linked List 706

EXAMPLE Template Version of Linked List Tools 711**17.2 LINKED LIST APPLICATIONS 715****EXAMPLE** A Stack Template Class 715**EXAMPLE** A Queue Template Class 722**TIP** A Comment on Namespaces 725

Friend Classes and Similar Alternatives 726

17.3 ITERATORS 729

Pointers as Iterators 729

Iterator Classes 730

EXAMPLE An Iterator Class 731**17.4 TREES 738**

Tree Properties 739

EXAMPLE A Tree Template Class 742**CHAPTER SUMMARY** 749**ANSWERS TO SELF-TEST EXERCISES** 747**PROGRAMMING PROJECTS** 754**Chapter 18 Exception Handling 757****18.1 EXCEPTION HANDLING BASICS 759**

A Toy Example of Exception Handling 759

Defining Your Own Exception Classes 768

Multiple Throws and Catches 768

PITFALL Catch the More Specific Exception First 772
TIP Exception Classes Can Be Trivial 773
Throwing an Exception in a Function 773
Exception Specification 775
PITFALL Exception Specification in Derived Classes 777

18.2 PROGRAMMING TECHNIQUES FOR EXCEPTION HANDLING 779

When to Throw an Exception 779

PITFALL Uncaught Exceptions 781

PITFALL Nested try-catch Blocks 781

PITFALL Overuse of Exceptions 782

Exception Class Hierarchies 782

Testing for Available Memory 782

Rethrowing an Exception 783

CHAPTER SUMMARY 784

ANSWERS TO SELF-TEST EXERCISES 784

PROGRAMMING PROJECTS 785

Chapter 19 Standard Template Library 787

19.1 ITERATORS 789

Iterator Basics 789

Kinds of Iterators 795

Constant and Mutable Iterators 798

Reverse Iterators 800

PITFALL Compiler Problems 802

Other Kinds of Iterators 802

19.2 CONTAINERS 803

Sequential Containers 803

PITFALL Iterators and Removing Elements 808

TIP Type Definitions in Containers 808

The Container Adapters `stack` and `queue` 809

The Associative Containers `set` and `map` 810

Efficiency 815

19.3 GENERIC ALGORITHMS 817

Running Times and Big-O Notation 818

Container Access Running Times 822

Nonmodifying Sequence Algorithms 823

Modifying Sequence Algorithms 828

Set Algorithms 828

Sorting Algorithms 829

CHAPTER SUMMARY	831
ANSWERS TO SELF-TEST EXERCISES	832
PROGRAMMING PROJECTS	833

Chapter 20 Patterns and UML 837

20.1 PATTERNS	838
Adapter Pattern	839
The Model-View-Controller Pattern	839
EXAMPLE A Sorting Pattern	841
Efficiency of the Sorting Pattern	845
TIP Pragmatics and Patterns	847
Pattern Formalism	848
20.2 UML	849
History of UML	849
UML Class Diagrams	850
Class Interactions	850

CHAPTER SUMMARY	851
ANSWERS TO SELF-TEST EXERCISES	851
PROGRAMMING PROJECTS	853

Appendix 1 C++ Keywords	855
Appendix 2 Precedence of Operators	857
Appendix 3 The ASCII Character Set	859
Appendix 4 Some Library Functions	861
Arithmetic Functions	861
Input and Output Member Functions	862
Character Functions	863
C-String Functions	864
string Class Functions	866
Random Number Generator	867
Trigonometric Functions	868
Appendix 5 Old and New Header Files	869
Further Reading	871
Index	873