

Tenth Edition  
Global Edition

**PROBLEM SOLVING**

with

**C++**

**Walter J. Savitch**

*UNIVERSITY OF CALIFORNIA, SAN DIEGO*

CONTRIBUTOR

**Kenrick Mock**

*UNIVERSITY OF ALASKA, ANCHORAGE*



330 Hudson Street, New York, NY 10013

Senior Vice President Courseware Portfolio Management:	Marcia J. Horton
Director, Portfolio Management: Engineering,	
Computer Science & Global Editions:	Julian Partridge
Portfolio Manager:	Matt Goldstein
Assistant Acquisitions Editor, Global Edition:	Aditee Agarwal
Portfolio Management Assistant:	Kristy Alaura
Field Marketing Manager:	Demetrius Hall
Product Marketing Manager:	Yvonne Vannatta
Managing Producer, ECS and Math:	Scott Disanno
Content Producer:	Sandra L. Rodriguez
Project Editor, Global Edition:	K.K. Neelakantan
Senior Manufacturing Controller, Global Edition:	Angela Hawksbee
Manager, Media Production, Global Edition:	Vikram Kumar
Cover Designer:	Lumina Datamatics, Inc.
Cover Photo:	Iana Chyrva/Shutterstock

The author and publisher of this book have used their best efforts in preparing this book. These efforts include the development, research, and testing of theories and programs to determine their effectiveness. The author and publisher make no warranty of any kind, expressed or implied, with regard to these programs or the documentation contained in this book. The author and publisher shall not be liable in any event for incidental or consequential damages with, or arising out of, the furnishing, performance, or use of these programs.

Pearson Education Limited  
 KAO Two  
 KAO Park  
 Harlow  
 CM17 9NA  
 United Kingdom

and Associated Companies throughout the world

Visit us on the World Wide Web at: [www.pearsonglobaleditions.com](http://www.pearsonglobaleditions.com)

© Pearson Education Limited 2018

The rights of Walter Savitch to be identified as the author of this work has been asserted by him in accordance with the Copyright, Designs and Patents Act 1988.

*Authorized adaptation from the United States edition, entitled Problem Solving with C++, 10th Edition, ISBN 978-0-13-444828-2 by Walter Savitch published by Pearson Education © 2018.*

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 either the prior written permission of the publisher or a license permitting restricted copying in the United Kingdom issued by the Copyright Licensing Agency Ltd, Saffron House, 6–10 Kirby Street, London EC1N 8TS.

All trademarks used herein are the property of their respective owners. The use of any trademark in this text does not vest in the author or publisher any trademark ownership rights in such trademarks, nor does the use of such trademarks imply any affiliation with or endorsement of this book by such owners.

British Library Cataloguing-in-Publication Data

A catalogue record for this book is available from the British Library

10 9 8 7 6 5 4 3 2 1

ISBN 10: 1-292-22282-4

ISBN 13: 978-1-292-22282-0

Typeset by iEnergizer Aptara®, Ltd.

Printed and bound in Malaysia

# Brief Contents

---

<b>Chapter 1</b>	<b>Introduction to Computers and C++ Programming</b>	<b>33</b>
<b>Chapter 2</b>	<b>C++ Basics</b>	<b>71</b>
<b>Chapter 3</b>	<b>More Flow of Control</b>	<b>143</b>
<b>Chapter 4</b>	<b>Procedural Abstraction and Functions That Return a Value</b>	<b>213</b>
<b>Chapter 5</b>	<b>Functions for All Subtasks</b>	<b>283</b>
<b>Chapter 6</b>	<b>I/O Streams as an Introduction to Objects and Classes</b>	<b>339</b>
<b>Chapter 7</b>	<b>Arrays</b>	<b>411</b>
<b>Chapter 8</b>	<b>Strings and Vectors</b>	<b>485</b>
<b>Chapter 9</b>	<b>Pointers and Dynamic Arrays</b>	<b>541</b>
<b>Chapter 10</b>	<b>Defining Classes</b>	<b>575</b>
<b>Chapter 11</b>	<b>Friends, Overloaded Operators, and Arrays in Classes</b>	<b>653</b>
<b>Chapter 12</b>	<b>Separate Compilation and Namespaces</b>	<b>737</b>

<b>Chapter 13</b>	<b>Pointers and Linked Lists</b>	773
<b>Chapter 14</b>	<b>Recursion</b>	823
<b>Chapter 15</b>	<b>Inheritance</b>	867
<b>Chapter 16</b>	<b>Exception Handling</b>	927
<b>Chapter 17</b>	<b>Templates</b>	959
<b>Chapter 18</b>	<b>Standard Template Library and C++11</b>	991
<b>Appendices</b>		
<b>1</b>	<b>C++ Keywords</b>	1067
<b>2</b>	<b>Precedence of Operators</b>	1068
<b>3</b>	<b>The ASCII Character Set</b>	1070
<b>4</b>	<b>Some Library Functions</b>	1071
<b>5</b>	<b>Inline Functions</b>	1078
<b>6</b>	<b>Overloading the Array Index Square Brackets</b>	1079
<b>7</b>	<b>The this Pointer</b>	1081
<b>8</b>	<b>Overloading Operators as Member Operators</b>	1084
<b>Credits</b>		1086
<b>Index</b>		1089

# Contents

---

## Chapter 1 Introduction to Computers and C++ Programming 33

### 1.1 COMPUTER SYSTEMS 34

Hardware 34

Software 39

High-Level Languages 40

Compilers 41

*History Note* 44

### 1.2 PROGRAMMING AND PROBLEM-SOLVING 44

Algorithms 44

Program Design 47

Object-Oriented Programming 48

The Software Life Cycle 49

### 1.3 INTRODUCTION TO C++ 50

Origins of the C++ Language 50

A Sample C++ Program 51

*Pitfall:* Using the Wrong Slash in `\n` 55

*Programming Tip:* Input and Output Syntax 55

Layout of a Simple C++ Program 56

*Pitfall:* Putting a Space Before the `include` File Name 58

Compiling and Running a C++ Program 58

*Pitfall:* Compiling a C++11 Program 59

*Programming Tip:* Getting Your Program to Run 59

### 1.4 TESTING AND DEBUGGING 61

Kinds of Program Errors 62

*Pitfall:* Assuming Your Program Is Correct 63

Chapter Summary 64

Answers to Self-Test Exercises 65

Practice Programs 67

Programming Projects 68

## Chapter 2 C++ Basics 71

### 2.1 VARIABLES AND ASSIGNMENTS 72

Variables 72

Names: Identifiers 74

Variable Declarations 77

Assignment Statements 77

*Pitfall:* Uninitialized Variables 79

*Programming Tip:* Use Meaningful Names 81

### 2.2 INPUT AND OUTPUT 82

Output Using `cout` 82

Include Directives and Namespaces 84

Escape Sequences 85

*Programming Tip:* End Each Program with a `\n` or `endl` 87

Formatting for Numbers with a Decimal Point 87

Input Using `cin` 88

Designing Input and Output 90

*Programming Tip:* Line Breaks in I/O 90

### 2.3 DATA TYPES AND EXPRESSIONS 92

The Types `int` and `double` 92

Other Number Types 94

C++11 Types 95

The Type `char` 96

The Type `bool` 98

Introduction to the Class `string` 98

Type Compatibilities 100

Arithmetic Operators and Expressions 101

*Pitfall:* Whole Numbers in Division 104

More Assignment Statements 106

### 2.4 SIMPLE FLOW OF CONTROL 106

A Simple Branching Mechanism 107

*Pitfall:* Strings of Inequalities 112

*Pitfall:* Using `=` in place of `==` 113

Compound Statements 114

Simple Loop Mechanisms 116

Increment and Decrement Operators 119

*Programming Example:* Charge Card Balance 121

*Pitfall:* Infinite Loops 122

## 2.5 PROGRAM STYLE 125

Indenting 125

Comments 125

Naming Constants 127

Chapter Summary 130

Answers to Self-Test Exercises 130

Practice Programs 135

Programming Projects 137

## Chapter 3 More Flow of Control 143

### 3.1 USING BOOLEAN EXPRESSIONS 144

Evaluating Boolean Expressions 144

*Pitfall:* Boolean Expressions Convert to *int* Values 148

Enumeration Types (*Optional*) 151

### 3.2 MULTIWAY BRANCHES 152

Nested Statements 152

*Programming Tip:* Use Braces in Nested Statements 153

Multiway *if-else* Statements 155

*Programming Example:* State Income Tax 157

The *switch* Statement 160

*Pitfall:* Forgetting a *break* in a *switch* Statement 164

Using *switch* Statements for Menus 165

Blocks 167

*Pitfall:* Inadvertent Local Variables 170

### 3.3 MORE ABOUT C++ LOOP STATEMENTS 171

The *while* Statements Reviewed 171

Increment and Decrement Operators Revisited 173

The *for* Statement 176

*Pitfall:* Extra Semicolon in a *for* Statement 181

What Kind of Loop to Use 182

*Pitfall:* Uninitialized Variables and Infinite Loops 184

The *break* Statement 185

*Pitfall:* The *break* Statement in Nested Loops 186

### 3.4 DESIGNING LOOPS 187

Loops for Sums and Products 187

Ending a Loop 189

Nested Loops	192
Debugging Loops	194
Chapter Summary	197
Answers to Self-Test Exercises	198
Practice Programs	204
Programming Projects	206

## **Chapter 4** Procedural Abstraction and Functions That Return a Value 213

### **4.1 TOP-DOWN DESIGN 214**

### **4.2 PREDEFINED FUNCTIONS 215**

Using Predefined Functions	215
Random Number Generation	220
Type Casting	222
Older Form of Type Casting	224
<i>Pitfall: Integer Division Drops the Fractional Part</i>	224

### **4.3 PROGRAMMER-DEFINED FUNCTIONS 225**

Function Definitions	225
Functions That Return a Boolean Value	231
Alternate Form for Function Declarations	231
<i>Pitfall: Arguments in the Wrong Order</i>	232
Function Definition–Syntax Summary	233
More About Placement of Function Definitions	234
<i>Programming Tip: Use Function Calls in Branching Statements</i>	235

### **4.4 PROCEDURAL ABSTRACTION 236**

The Black-Box Analogy	236
<i>Programming Tip: Choosing Formal Parameter Names</i>	239
<i>Programming Tip: Nested Loops</i>	240
<i>Case Study: Buying Pizza</i>	243
<i>Programming Tip: Use Pseudocode</i>	249

### **4.5 SCOPE AND LOCAL VARIABLES 250**

The Small Program Analogy	250
<i>Programming Example: Experimental Pea Patch</i>	253
Global Constants and Global Variables	253
Call-by-Value Formal Parameters Are Local Variables	256
Block Scope	258



Namespaces Revisited	259
<i>Programming Example: The Factorial Function</i>	262
<b>4.6 OVERLOADING FUNCTION NAMES</b>	<b>264</b>
Introduction to Overloading	264
<i>Programming Example: Revised Pizza-Buying Program</i>	267
Automatic Type Conversion	270
Chapter Summary	272
Answers to Self-Test Exercises	272
Practice Programs	277
Programming Projects	279

## **Chapter 5 Functions for All Subtasks** 283

<b>5.1 VOID FUNCTIONS</b>	<b>284</b>
Definitions of <i>void</i> Functions	284
<i>Programming Example: Converting Temperatures</i>	287
return Statements in <i>void</i> Functions	287
<b>5.2 CALL-BY-REFERENCE PARAMETERS</b>	<b>291</b>
A First View of Call-by-Reference	291
Call-by-Reference in Detail	294
<i>Programming Example: The swapValues Function</i>	299
Mixed Parameter Lists	300
<i>Programming Tip: What Kind of Parameter to Use</i>	301
<i>Pitfall: Inadvertent Local Variables</i>	302
<b>5.3 USING PROCEDURAL ABSTRACTION</b>	<b>305</b>
Functions Calling Functions	305
Preconditions and Postconditions	307
<i>Case Study: Supermarket Pricing</i>	308
<b>5.4 TESTING AND DEBUGGING FUNCTIONS</b>	<b>313</b>
Stubs and Drivers	314
<b>5.5 GENERAL DEBUGGING TECHNIQUES</b>	<b>319</b>
Keep an Open Mind	319
Check Common Errors	319
Localize the Error	320
The <code>assert</code> Macro	322

Chapter Summary	324
Answers to Self-Test Exercises	325
Practice Programs	328
Programming Projects	331

## Chapter 6 I/O Streams as an Introduction to Objects and Classes 339

### 6.1 STREAMS AND BASIC FILE I/O 340

Why Use Files for I/O?	341
File I/O	342
Introduction to Classes and Objects	346
<i>Programming Tip: Check Whether a File Was Opened Successfully</i>	348
Techniques for File I/O	350
Appending to a File ( <i>Optional</i> )	354
File Names as Input ( <i>Optional</i> )	355

### 6.2 TOOLS FOR STREAM I/O 357

Formatting Output with Stream Functions	357
Manipulators	363
Streams as Arguments to Functions	366
<i>Programming Tip: Checking for the End of a File</i>	366
A Note on Namespaces	369
<i>Programming Example: Cleaning Up a File Format</i>	370

### 6.3 CHARACTER I/O 372

The Member Functions <code>get</code> and <code>put</code>	372
The <code>putback</code> Member Function ( <i>Optional</i> )	376
<i>Programming Example: Checking Input</i>	377
<i>Pitfall: Unexpected '\n' in Input</i>	379
<i>Programming Example: Another <code>newLine</code> Function</i>	381
Default Arguments for Functions ( <i>Optional</i> )	382
The <code>eof</code> Member Function	387
<i>Programming Example: Editing a Text File</i>	389
Predefined Character Functions	390
<i>Pitfall: <code>toupper</code> and <code>tolower</code> Return Values</i>	392

Chapter Summary	394
Answers to Self-Test Exercises	395
Practice Programs	402
Programming Projects	404

## Chapter 7 Arrays 411

### 7.1 INTRODUCTION TO ARRAYS 412

Declaring and Referencing Arrays 412

*Programming Tip:* Use *for* Loops with Arrays 414

*Pitfall:* Array Indexes Always Start with Zero 414

*Programming Tip:* Use a Defined *Constant* for the Size of  
an Array 414

Arrays in Memory 416

*Pitfall:* Array Index Out of Range 417

Initializing Arrays 420

*Programming Tip:* C++11 Range-Based *for* Statement 420

### 7.2 ARRAYS IN FUNCTIONS 423

Indexed Variables as Function Arguments 423

Entire Arrays as Function Arguments 425

The *const* Parameter Modifier 428

*Pitfall:* Inconsistent Use of *const* Parameters 431

Functions That Return an Array 431

*Case Study:* Production Graph 432

### 7.3 PROGRAMMING WITH ARRAYS 445

Partially Filled Arrays 445

*Programming Tip:* Do Not Skimp on Formal Parameters 448

*Programming Example:* Searching an Array 448

*Programming Example:* Sorting an Array 451

*Programming Example:* Bubble Sort 455

### 7.4 MULTIDIMENSIONAL ARRAYS 458

Multidimensional Array Basics 459

Multidimensional Array Parameters 459

*Programming Example:* Two-Dimensional Grading  
Program 461

*Pitfall:* Using Commas Between Array Indexes 465

Chapter Summary 466

Answers to Self-Test Exercises 467

Practice Programs 471

Programming Projects 473

## Chapter 8 Strings and Vectors 485

### 8.1 AN ARRAY TYPE FOR STRINGS 487

C-String Values and C-String Variables 487

*Pitfall:* Using = and == with C Strings 490

Other Functions in `<cstring>` 492

*Pitfall:* Copying past the end of a C-string using `strcpy` 495

C-String Input and Output 498

C-String-to-Number Conversions and Robust Input 500

### 8.2 THE STANDARD STRING CLASS 506

Introduction to the Standard Class `string` 506

I/O with the Class `string` 509

*Programming Tip:* More Versions of `getline` 512

*Pitfall:* Mixing `cin >> variable;` and `getline` 513

String Processing with the Class `string` 514

*Programming Example:* Palindrome Testing 518

Converting between `string` Objects and C Strings 521

Converting Between Strings and Numbers 522

### 8.3 VECTORS 523

Vector Basics 523

*Pitfall:* Using Square Brackets Beyond the Vector Size 526

*Programming Tip:* Vector Assignment Is Well Behaved 527

Efficiency Issues 527

Chapter Summary 529

Answers to Self-Test Exercises 529

Practice Programs 531

Programming Projects 532

## Chapter 9 Pointers and Dynamic Arrays 541

### 9.1 POINTERS 542

Pointer Variables 543

Basic Memory Management 550

*Pitfall:* Dangling Pointers 551

Static Variables and Automatic Variables 552

*Programming Tip:* Define Pointer Types 552

## 9.2 DYNAMIC ARRAYS 555

- Array Variables and Pointer Variables 555
- Creating and Using Dynamic Arrays 556
- Pointer Arithmetic (*Optional*) 562
- Multidimensional Dynamic Arrays (*Optional*) 564
- Chapter Summary 566
- Answers to Self-Test Exercises 566
- Practice Programs 567
- Programming Projects 568

## Chapter 10 Defining Classes 575

### 10.1 STRUCTURES 576

- Structures for Diverse Data 576
- Pitfall*: Forgetting a Semicolon in a Structure Definition 581
- Structures as Function Arguments 582
- Programming Tip*: Use Hierarchical Structures 583
- Initializing Structures 585

### 10.2 CLASSES 588

- Defining Classes and Member Functions 588
- Public and Private Members 593
- Programming Tip*: Make All Member Variables Private 601
- Programming Tip*: Define Accessor and Mutator Functions 601
- Programming Tip*: Use the Assignment Operator with Objects 603
- Programming Example*: BankAccount Class—Version 1 604
- Summary of Some Properties of Classes 608
- Constructors for Initialization 610
- Programming Tip*: Always Include a Default Constructor 618
- Pitfall*: Constructors with No Arguments 619
- Member Initializers and Constructor Delegation in C++11 621

### 10.3 ABSTRACT DATA TYPES 622

- Classes to Produce Abstract Data Types 623
- Programming Example*: Alternative Implementation of a Class 627

### 10.4 INTRODUCTION TO INHERITANCE 632

- Derived Classes 633
- Defining Derived Classes 634

Chapter Summary	638
Answers to Self-Test Exercises	639
Practice Programs	645
Programming Projects	646

## **Chapter 11** Friends, Overloaded Operators, and Arrays in Classes 653

### **11.1 FRIEND FUNCTIONS 654**

<i>Programming Example: An Equality Function</i>	654
Friend Functions	658
<i>Programming Tip: Define Both Accessor Functions and Friend Functions</i>	660
<i>Programming Tip: Use Both Member and Nonmember Functions</i>	662
<i>Programming Example: Money Class (Version 1)</i>	662
Implementation of <code>digitToInt</code> ( <i>Optional</i> )	669
<i>Pitfall: Leading Zeros in Number Constants</i>	670
The <code>const</code> Parameter Modifier	672
<i>Pitfall: Inconsistent Use of <code>const</code></i>	673

### **11.2 OVERLOADING OPERATORS 677**

Overloading Operators	678
Constructors for Automatic Type Conversion	681
Overloading Unary Operators	683
Overloading <code>&gt;&gt;</code> and <code>&lt;&lt;</code>	684

### **11.3 ARRAYS AND CLASSES 694**

Arrays of Classes	694
Arrays as Class Members	698
<i>Programming Example: A Class for a Partially Filled Array</i>	699

### **11.4 CLASSES AND DYNAMIC ARRAYS 701**

<i>Programming Example: A String Variable Class</i>	702
Destructors	705
<i>Pitfall: Pointers as Call-by-Value Parameters</i>	708
Copy Constructors	709
Overloading the Assignment Operator	714
Chapter Summary	717
Answers to Self-Test Exercises	717
Practice Programs	727
Programming Projects	728

## Chapter 12 Separate Compilation and Namespaces 737

### 12.1 SEPARATE COMPILATION 738

ADTs Reviewed 739

*Case Study:* DigitalTime—A Class Compiled Separately 740

Using `#ifndef` 749

*Programming Tip:* Defining Other Libraries 752

### 12.2 NAMESPACES 753

Namespaces and *using* Directives 754

Creating a Namespace 755

Qualifying Names 758

A Subtle Point About Namespaces (*Optional*) 759

Unnamed Namespaces 760

*Programming Tip:* Choosing a Name for a Namespace 765

*Pitfall:* Confusing the Global Namespace and the Unnamed Namespace 766

Chapter Summary 767

Answers to Self-Test Exercises 768

Practice Programs 770

Programming Projects 772

## Chapter 13 Pointers and Linked Lists 773

### 13.1 NODES AND LINKED LISTS 774

Nodes 774

`nullptr` 779

Linked Lists 780

Inserting a Node at the Head of a List 781

*Pitfall:* Losing Nodes 784

Searching a Linked List 785

Pointers as Iterators 789

Inserting and Removing Nodes Inside a List 789

*Pitfall:* Using the Assignment Operator with Dynamic Data Structures 791

Variations on Linked Lists 794

Linked Lists of Classes 796

### 13.2 STACKS AND QUEUES 799

Stacks 799

*Programming Examples:* A Stack Class 800

Queues 805

*Programming Examples:* A Queue Class 806

Chapter Summary	810
Answers to Self-Test Exercises	810
Practice Programs	813
Programming Projects	814

## Chapter 14 Recursion 823

### 14.1 RECURSIVE FUNCTIONS FOR TASKS 825

<i>Case Study:</i> Vertical Numbers	825
A Closer Look at Recursion	831
<i>Pitfall:</i> Infinite Recursion	833
Stacks for Recursion	834
<i>Pitfall:</i> Stack Overflow	836
Recursion Versus Iteration	836

### 14.2 RECURSIVE FUNCTIONS FOR VALUES 838

General Form for a Recursive Function That Returns a Value	838
<i>Programming Example:</i> Another Powers Function	838

### 14.3 THINKING RECURSIVELY 843

Recursive Design Techniques	843
<i>Case Study:</i> Binary Search—An Example of Recursive Thinking	844
<i>Programming Example:</i> A Recursive Member Function	852

Chapter Summary	856
Answers to Self-Test Exercises	856
Practice Programs	861
Programming Projects	861

## Chapter 15 Inheritance 867

### 15.1 INHERITANCE BASICS 868

Derived Classes	871
Constructors in Derived Classes	879
<i>Pitfall:</i> Use of Private Member Variables from the Base Class	882
<i>Pitfall:</i> Private Member Functions Are Effectively Not Inherited	884
The <i>protected</i> Qualifier	884
Redefinition of Member Functions	887
Redefining Versus Overloading	890
Access to a Redefined Base Function	892

### 15.2 INHERITANCE DETAILS 893

Functions That Are Not Inherited	893
----------------------------------	-----



Assignment Operators and Copy Constructors in Derived Classes	894
Destructors in Derived Classes	895
<b>15.3 POLYMORPHISM</b>	<b>896</b>
Late Binding	897
Virtual Functions in C++	898
Virtual Functions and Extended Type Compatibility	903
<i>Pitfall</i> : The Slicing Problem	907
<i>Pitfall</i> : Not Using Virtual Member Functions	908
<i>Pitfall</i> : Attempting to Compile Class Definitions Without Definitions for Every Virtual Member Function	909
<i>Programming Tip</i> : Make Destructors Virtual	909
Chapter Summary	911
Answers to Self-Test Exercises	911
Practice Programs	915
Programming Projects	918
<b>Chapter 16 Exception Handling</b>	<b>927</b>
<b>16.1 EXCEPTION-HANDLING BASICS</b>	<b>929</b>
A Toy Example of Exception Handling	929
Defining Your Own Exception Classes	938
Multiple Throws and Catches	938
<i>Pitfall</i> : Catch the More Specific Exception First	942
<i>Programming Tip</i> : Exception Classes Can Be Trivial	943
Throwing an Exception in a Function	943
Exception Specification	945
<i>Pitfall</i> : Exception Specification in Derived Classes	947
<b>16.2 PROGRAMMING TECHNIQUES FOR EXCEPTION HANDLING</b>	<b>948</b>
When to Throw an Exception	948
<i>Pitfall</i> : Uncaught Exceptions	950
<i>Pitfall</i> : Nested <i>try-catch</i> Blocks	950
<i>Pitfall</i> : Overuse of Exceptions	950
Exception Class Hierarchies	951
Testing for Available Memory	951
Rethrowing an Exception	952
Chapter Summary	952
Answers to Self-Test Exercises	952
Practice Programs	954
Programming Projects	955

## Chapter 17 Templates 959

### 17.1 TEMPLATES FOR ALGORITHM ABSTRACTION 960

Templates for Functions 961

*Pitfall:* Compiler Complications 965

*Programming Example:* A Generic Sorting Function 967

*Programming Tip:* How to Define Templates 971

*Pitfall:* Using a Template with an Inappropriate Type 972

### 17.2 TEMPLATES FOR DATA ABSTRACTION 973

Syntax for Class Templates 973

*Programming Example:* An Array Class 976

Chapter Summary 983

Answers to Self-Test Exercises 983

Practice Programs 987

Programming Projects 987

## Chapter 18 Standard Template Library and C++11 991

### 18.1 ITERATORS 993

using Declarations 993

Iterator Basics 994

*Programming Tip:* Use auto to Simplify Variable Declarations 998

*Pitfall:* Compiler Problems 998

Kinds of Iterators 1000

Constant and Mutable Iterators 1004

Reverse Iterators 1005

Other Kinds of Iterators 1006

### 18.2 CONTAINERS 1007

Sequential Containers 1008

*Pitfall:* Iterators and Removing Elements 1012

*Programming Tip:* Type Definitions in Containers 1013

Container Adapters `stack` and `queue` 1013

Associative Containers `set` and `map` 1017

*Programming Tip:* Use Initialization, Ranged `for`, and `auto` with Containers 1024

Efficiency 1024

### 18.3 GENERIC ALGORITHMS 1025

Running Times and Big-O Notation 1026

Container Access Running Times 1029

Nonmodifying Sequence Algorithms 1031  
Container Modifying Algorithms 1035  
Set Algorithms 1037  
Sorting Algorithms 1038

#### 18.4 C++ IS EVOLVING 1039

std::array 1039  
Regular Expressions 1040  
Threads 1045  
Smart Pointers 1051  
  
Chapter Summary 1057  
Answers to Self-Test Exercises 1058  
Practice Programs 1059  
Programming Projects 1061

#### APPENDICES

- 1 C++ Keywords 1067
- 2 Precedence of Operators 1068
- 3 The ASCII Character Set 1070
- 4 Some Library Functions 1071
- 5 Inline Functions 1078
- 6 Overloading the Array Index Square Brackets 1079
- 7 The this Pointer 1081
- 8 Overloading Operators as Member Operators 1084

CREDITS 1086

INDEX 1089