

# ABSOLUTE

5TH EDITION C++

Walter Savitch

*University of California, San Diego*

Contributor

Kenrick Mock

*University of Alaska Anchorage*

PEARSON

Boston Columbus Indianapolis New York San Francisco Upper Saddle River  
Amsterdam Cape Town Dubai London Madrid Milan Munich Paris Montréal Toronto  
Delhi Mexico City São Paulo Sydney Hong Kong Seoul Singapore Taipei Tokyo

# Brief Contents

Chapter 1	C++ BASICS	1
Chapter 2	FLOW OF CONTROL	45
Chapter 3	FUNCTION BASICS	99
Chapter 4	PARAMETERS AND OVERLOADING	145
Chapter 5	ARRAYS	185
Chapter 6	STRUCTURES AND CLASSES	239
Chapter 7	CONSTRUCTORS AND OTHER TOOLS	275
Chapter 8	OPERATOR OVERLOADING, FRIENDS, AND REFERENCES	321
Chapter 9	STRINGS	367
Chapter 10	POINTERS AND DYNAMIC ARRAYS	419
Chapter 11	SEPARATE COMPILATION AND NAMESPACES	471
Chapter 12	STREAMS AND FILE I/O	515
Chapter 13	RECURSION	571
Chapter 14	INHERITANCE	613
Chapter 15	POLYMORPHISM AND VIRTUAL FUNCTIONS	661
Chapter 16	TEMPLATES	693
Chapter 17	LINKED DATA STRUCTURES	731
Chapter 18	EXCEPTION HANDLING	825
Chapter 19	STANDARD TEMPLATE LIBRARY	857
Chapter 20	PATTERNS AND UML (online at <a href="http://www.pearsonhighered.com/savitch">www.pearsonhighered.com/savitch</a> )	
Appendix 1	C++ KEYWORDS	915
Appendix 2	PRECEDENCE OF OPERATORS	917
Appendix 3	THE ASCII CHARACTER SET	919
Appendix 4	SOME LIBRARY FUNCTIONS	921
Appendix 5	OLD AND NEW HEADER FILES	929
	INDEX	931

# 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 7
- Variables 8
- Assignment Statements 10
- Introduction to the `string` class 12
- PITFALL: Uninitialized Variables 12
- TIP: Use Meaningful Names 13
- More Assignment Statements 14
- Assignment Compatibility 15
- Literals 16
- 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 35

**1.4 PROGRAM STYLE 36**

Comments 36

**1.5 LIBRARIES AND NAMESPACES 37**Libraries and `include` Directives 37

Namespaces 37

PITFALL: Problems with Library Names 38

Chapter Summary 39

Answers to Self-Test Exercises 40

Programming Projects 42

**Chapter 2 Flow of Control 45****2.1 BOOLEAN EXPRESSIONS 45**

Building Boolean Expressions 46

PITFALL: Strings of Inequalities 47

Evaluating Boolean Expressions 48

Precedence Rules 50

PITFALL: Integer Values Can Be Used as Boolean Values 54

**2.2 BRANCHING MECHANISMS 56**`if-else` Statements 56

Compound Statements 58

PITFALL: Using `=` in Place of `==` 59Omitting the `else` 61

Nested Statements 61

Multiway `if-else` Statement 61The `switch` Statement 62PITFALL: Forgetting a `break` in a `switch` Statement 65TIP: Use `switch` Statements for Menus 65

Enumeration Types 66

The Conditional Operator 66

**2.3 LOOPS 67**The `while` and `do-while` Statements 68

Increment and Decrement Operators Revisited 71

The Comma Operator 72

The `for` Statement 74TIP: Repeat-*N*-Times Loops 76PITFALL: Extra Semicolon in a `for` Statement 77

PITFALL: Infinite Loops 77

The `break` and `continue` Statements 80

Nested Loops 83

<b>2.4</b>	<b>INTRODUCTION TO FILE INPUT</b>	<b>83</b>
	Reading From a Text File Using <code>ifstream</code>	84
	Chapter Summary	87
	Answers to Self-Test Exercises	87
	Programming Projects	93
<b>Chapter 3</b>	<b>Function Basics</b>	<b>99</b>
<b>3.1</b>	<b>PREDEFINED FUNCTIONS</b>	<b>100</b>
	Predefined Functions That Return a Value	100
	Predefined <code>void</code> Functions	105
	A Random Number Generator	107
<b>3.2</b>	<b>PROGRAMMER-DEFINED FUNCTIONS</b>	<b>111</b>
	Defining Functions That Return a Value	112
	Alternate Form for Function Declarations	114
	PITFALL: Arguments in the Wrong Order	115
	PITFALL: Use of the Terms <i>Parameter</i> and <i>Argument</i>	115
	Functions Calling Functions	115
	EXAMPLE: A Rounding Function	115
	Functions That Return a Boolean Value	118
	Defining <code>void</code> Functions	119
	<code>return</code> Statements in <code>void</code> Functions	121
	Preconditions and Postconditions	121
	<code>main</code> Is a Function	123
	Recursive Functions	123
<b>3.3</b>	<b>SCOPE RULES</b>	<b>125</b>
	Local Variables	125
	Procedural Abstraction	127
	Global Constants and Global Variables	128
	Blocks	131
	Nested Scopes	132
	TIP: Use Function Calls in Branching and Loop Statements	132
	Variables Declared in a <code>for</code> Loop	133
	Chapter Summary	134
	Answers to Self-Test Exercises	134
	Programming Projects	138

## Chapter 4 Parameters and Overloading 145

### 4.1 PARAMETERS 146

- Call-by-Value Parameters 146
- A First Look at Call-by-Reference Parameters 148
- Call-by-Reference Mechanism in Detail 151
- Constant Reference Parameters 153
- EXAMPLE: The `swapValues` Function 153
- TIP: Think of Actions, Not Code 154
- Mixed Parameter Lists 155
- TIP: What Kind of Parameter to Use 156
- PITFALL: Inadvertent Local Variables 158
- TIP: Choosing Formal Parameter Names 159
- EXAMPLE: Buying Pizza 160

### 4.2 OVERLOADING AND DEFAULT ARGUMENTS 163

- Introduction to Overloading 163
- PITFALL: Automatic Type Conversion and Overloading 166
- Rules for Resolving Overloading 167
- EXAMPLE: Revised Pizza-Buying Program 169
- Default Arguments 171

### 4.3 TESTING AND DEBUGGING FUNCTIONS 173

- The `assert` Macro 173
- Stubs and Drivers 174
- Chapter Summary 177
- Answers to Self-Test Exercises 177
- Programming Projects 179

## Chapter 5 Arrays 185

### 5.1 INTRODUCTION TO ARRAYS 186

- Declaring and Referencing Arrays 186
- TIP: Use `for` Loops with Arrays 189
- PITFALL: Array Indexes Always Start with Zero 189
- TIP: Use a Defined Constant for the Size of an Array 189
- Arrays in Memory 190
- PITFALL: Array Index out of Range 192
- Initializing Arrays 192

### 5.2 ARRAYS IN FUNCTIONS 195

- Indexed Variables as Function Arguments 195
- Entire Arrays as Function Arguments 196

The `const` Parameter Modifier 200  
PITFALL: Inconsistent Use of `const` Parameters 201  
Functions That Return an Array 202  
EXAMPLE: Production Graph 202

### 5.3 PROGRAMMING WITH ARRAYS 207

Partially Filled Arrays 207  
TIP: Do Not Skimp on Formal Parameters 208  
EXAMPLE: Searching an Array 211  
EXAMPLE: Sorting an Array 213

### 5.4 MULTIDIMENSIONAL ARRAYS 218

Multidimensional Array Basics 218  
Multidimensional Array Parameters 219  
EXAMPLE: Two-Dimensional Grading Program 220  
  
Chapter Summary 225  
Answers to Self-Test Exercises 226  
Programming Projects 230

## Chapter 6 Structures and Classes 239

### 6.1 STRUCTURES 240

Structure Types 242  
PITFALL: Forgetting a Semicolon in a Structure Definition 246  
Structures as Function Arguments 246  
TIP: Use Hierarchical Structures 247  
Initializing Structures 249

### 6.1 CLASSES 252

Defining Classes and Member Functions 252  
Encapsulation 258  
Public and Private Members 259  
Accessor and Mutator Functions 262  
TIP: Separate Interface and Implementation 264  
TIP: A Test for Encapsulation 265  
Structures versus Classes 266  
TIP: Thinking Objects 268  
  
Chapter Summary 268  
Answers to Self-Test Exercises 269  
Programming Projects 271

## Chapter 7 Constructors and Other Tools 275

### 7.1 CONSTRUCTORS 276

- Constructor Definitions 276
- PITFALL: Constructors with No Arguments 281
- Explicit Constructor Calls 282
- TIP: Always Include a Default Constructor 283
- EXAMPLE: `BankAccount` Class 285
- Class Type Member Variables 292

### 7.2 MORE TOOLS 295

- The `const` Parameter Modifier 295
- PITFALL: Inconsistent Use of `const` 297
- Inline Functions 301
- Static Members 303
- Nested and Local Class Definitions 306

### 7.3 VECTORS—A PREVIEW OF THE STANDARD TEMPLATE LIBRARY 307

- Vector Basics 307
- PITFALL: Using Square Brackets beyond the Vector Size 309
- TIP: Vector Assignment Is Well Behaved 311
- Efficiency Issues 311
- Chapter Summary 313
- Answers to Self-Test Exercises 313
- Programming Projects 315

## Chapter 8 Operator Overloading, Friends, and References 321

### 8.1 BASIC OPERATOR OVERLOADING 322

- Overloading Basics 323
- TIP: A Constructor Can Return an Object 328
- Returning by `const` Value 329
- Overloading Unary Operators 332
- Overloading as Member Functions 332
- TIP: A Class Has Access to All Its Objects 335
- Overloading Function Application `()` 335
- PITFALL: Overloading `&&`, `||`, and the Comma Operator 336

### 8.2 FRIEND FUNCTIONS AND AUTOMATIC TYPE CONVERSION 336

- Constructors for Automatic Type Conversion 336
- PITFALL: Member Operators and Automatic Type Conversion 337
- Friend Functions 338
- Friend Classes 341
- PITFALL: Compilers without Friends 342



## 8.3 REFERENCES AND MORE OVERLOADED OPERATORS 343

- References 344
- TIP: Returning Member Variables of a Class Type 345
- Overloading >> and << 346
- TIP: What Mode of Returned Value to Use 352
- The Assignment Operator 355
- Overloading the Increment and Decrement Operators 355
- Overloading the Array Operator [] 358
- Overloading Based on L-Value versus R-Value 360
- Chapter Summary 360
- Answers to Self-Test Exercises 361
- Programming Projects 363

## Chapter 9 Strings 367

### 9.1 AN ARRAY TYPE FOR STRINGS 368

- C-String Values and C-String Variables 369
- PITFALL: Using = and == with C-strings 372
- Other Functions in <cstring> 374
- EXAMPLE: Command-Line Arguments 376
- C-String Input and Output 379

### 9.2 CHARACTER MANIPULATION TOOLS 381

- Character I/O 381
- The Member Functions `get` and `put` 382
- EXAMPLE: Checking Input Using a Newline Function 384
- PITFALL: Unexpected '\n' in Input 386
- The `putback`, `peek`, and `ignore` Member Functions 387
- Character-Manipulating Functions 389
- PITFALL: `toupper` and `tolower` Return `int` Values 391

### 9.3 THE STANDARD CLASS `string` 393

- Introduction to the Standard Class `string` 393
- I/O with the Class `string` 396
- TIP: More Versions of `getline` 399
- PITFALL: Mixing `cin >> variable;` and `getline` 399
- String Processing with the Class `string` 401
- EXAMPLE: Palindrome Testing 404
- Converting between `string` Objects and C-Strings 408
- Chapter Summary 409
- Answers to Self-Test Exercises 409
- Programming Projects 413

<b>Chapter 10</b>	<b>Pointers and Dynamic Arrays</b>	<b>419</b>
<b>10.1</b>	<b>POINTERS</b>	<b>420</b>
Pointer Variables		421
Basic Memory Management		429
PITFALL: Dangling Pointers		432
Dynamic Variables and Automatic Variables		432
TIP: Define Pointer Types		433
PITFALL: Pointers as Call-by-Value Parameters		435
Uses for Pointers		436
<b>10.2</b>	<b>DYNAMIC ARRAYS</b>	<b>437</b>
Array Variables and Pointer Variables		437
Creating and Using Dynamic Arrays		439
EXAMPLE: A Function That Returns an Array		442
Pointer Arithmetic		444
Multidimensional Dynamic Arrays		445
<b>10.3</b>	<b>CLASSES, POINTERS, AND DYNAMIC ARRAYS</b>	<b>448</b>
The <code>-&gt;</code> Operator		448
The <code>this</code> Pointer		449
Overloading the Assignment Operator		449
EXAMPLE: A Class for Partially Filled Arrays		456
Destructors		459
Copy Constructors		460
Chapter Summary		465
Answers to Self-Test Exercises		465
Programming Projects		467
<b>Chapter 11</b>	<b>Separate Compilation and Namespaces</b>	<b>471</b>
<b>11.1</b>	<b>SEPARATE COMPILATION</b>	<b>472</b>
Encapsulation Reviewed		473
Header Files and Implementation Files		473
EXAMPLE: <code>DigitalTime</code> Class		482
TIP: Reusable Components		483
Using <code>#ifndef</code>		483
TIP: Defining Other Libraries		485
<b>11.2</b>	<b>NAMESPACES</b>	<b>487</b>
Namespaces and <code>using</code> Directives		487
Creating a Namespace		489
<code>using</code> Declarations		492

Qualifying Names	493
TIP: Choosing a Name for a Namespace	495
EXAMPLE: A Class Definition in a Namespace	496
Unnamed Namespaces	497
PITFALL: Confusing the Global Namespace and the Unnamed Namespace	503
TIP: Unnamed Namespaces Replace the <code>static</code> Qualifier	504
TIP: Hiding Helping Functions	504
Nested Namespaces	505
TIP: What Namespace Specification Should You Use?	505
Chapter Summary	508
Answers to Self-Test Exercises	508
Programming Projects	510

## Chapter 12 Streams and File I/O 515

### 12.1 I/O STREAMS 517

File I/O	517
PITFALL: Restrictions on Stream Variables	522
Appending to a File	522
TIP: Another Syntax for Opening a File	524
TIP: Check That a File Was Opened Successfully	526
Character I/O	528
Checking for the End of a File	529

### 12.2 TOOLS FOR STREAM I/O 533

File Names as Input	533
Formatting Output with Stream Functions	534
Manipulators	538
Saving Flag Settings	539
More Output Stream Member Functions	540
EXAMPLE: Cleaning Up a File Format	542
EXAMPLE: Editing a Text File	544

### 12.3 STREAM HIERARCHIES: A PREVIEW OF INHERITANCE 547

Inheritance among Stream Classes	547
EXAMPLE: Another <code>newLine</code> Function	549
Parsing Strings with the <code>stringstream</code> Class	553

### 12.4 RANDOM ACCESS TO FILES 556

Chapter Summary	558
Answers to Self-Test Exercises	558
Programming Projects	561

## Chapter 13 Recursion 571

### 13.1 RECURSIVE void FUNCTIONS 573

EXAMPLE: Vertical Numbers 573

Tracing a Recursive Call 576

A Closer Look at Recursion 579

PITFALL: Infinite Recursion 580

Stacks for Recursion 582

PITFALL: Stack Overflow 583

Recursion versus Iteration 584

### 13.2 RECURSIVE FUNCTIONS THAT RETURN A VALUE 585

General Form for a Recursive Function That Returns a Value 585

EXAMPLE: Another Powers Function 586

Mutual Recursion 591

### 13.3 THINKING RECURSIVELY 593

Recursive Design Techniques 593

Binary Search 594

Coding 596

Checking the Recursion 600

Efficiency 600

Chapter Summary 602

Answers to Self-Test Exercises 603

Programming Projects 607

## Chapter 14 Inheritance 613

### 14.1 INHERITANCE BASICS 614

Derived Classes 614

Constructors in Derived Classes 624

PITFALL: Use of Private Member Variables from the Base Class 626

PITFALL: Private Member Functions Are Effectively Not Inherited 628

The `protected` Qualifier 628

Redefinition of Member Functions 631

Redefining versus Overloading 632

Access to a Redefined Base Function 634

Functions That Are Not Inherited 635

### 14.2 PROGRAMMING WITH INHERITANCE 636

Assignment Operators and Copy Constructors in Derived Classes 636

Destructors in Derived Classes 637

EXAMPLE: Partially Filled Array with Backup 638

PITFALL: Same Object on Both Sides of the Assignment Operator 647

EXAMPLE: Alternate Implementation of <code>PfArrayDBak</code>	647
TIP: A Class Has Access to Private Members of All Objects of the Class	650
TIP: “Is a” versus “Has a”	650
Protected and Private Inheritance	651
Multiple Inheritance	652
Chapter Summary	653
Answers to Self-Test Exercises	653
Programming Projects	655

## Chapter 15 Polymorphism and Virtual Functions 661

### 15.1 VIRTUAL FUNCTION BASICS 662

Late Binding	662
Virtual Function in C++	663
TIP: The Virtual Property Is Inherited	669
TIP: When to Use a Virtual Function	670
PITFALL: Omitting the Definition of a Virtual Member Function	670
Abstract Classes and Pure Virtual Functions	671
EXAMPLE: An Abstract Class	672

### 15.2 POINTERS AND VIRTUAL FUNCTIONS 674

Virtual Functions and Extended Type Compatibility	674
PITFALL: The Slicing Problem	678
TIP: Make Destructors Virtual	679
Downcasting and Upcasting	680
How C++ Implements Virtual Functions	681
Chapter Summary	683
Answers to Self-Test Exercises	684
Programming Projects	684

## Chapter 16 Templates 693

### 16.1 FUNCTION TEMPLATES 694

Syntax for Function Templates	695
PITFALL: Compiler Complications	698
TIP: How to Define Templates	700
EXAMPLE: A Generic Sorting Function	701
PITFALL: Using a Template with an Inappropriate Type	705

### 16.2 CLASS TEMPLATES 707

Syntax for Class Templates	708
EXAMPLE: An Array Template Class	712
The <code>vector</code> and <code>basic_string</code> Templates	718

**16.3 TEMPLATES AND INHERITANCE 718**

EXAMPLE: Template Class For a Partially Filled Array with Backup 719

Chapter Summary 724

Answers to Self-Test Exercises 724

Programming Projects 728

**Chapter 17 Linked Data Structures 731****17.1 NODES AND LINKED LISTS 733**

Nodes 733

Linked Lists 738

Inserting a Node at the Head of a List 740

PITFALL: Losing Nodes 743

Inserting and Removing Nodes Inside a List 743

PITFALL: Using the Assignment Operator with Dynamic Data Structures 747

Searching a Linked List 747

Doubly Linked Lists 750

Adding a Node to a Doubly Linked List 752

Deleting a Node from a Doubly Linked List 752

EXAMPLE: A Generic Sorting Template Version of Linked List Tools 759

**17.2 LINKED LIST APPLICATIONS 763**

EXAMPLE: A Stack Template Class 763

EXAMPLE: A Queue Template Class 770

TIP: A Comment on Namespaces 773

Friend Classes and Similar Alternatives 774

EXAMPLE: Hash Tables With Chaining 777

Efficiency of Hash Tables 783

EXAMPLE: A Set Template Class 784

Efficiency of Sets Using Linked Lists 790

**17.3 ITERATORS 791**

Pointers as Iterators 792

Iterator Classes 792

EXAMPLE: An Iterator Class 794

**17.4 TREES 800**

Tree Properties 801

EXAMPLE: A Tree Template Class 803

Chapter Summary 808

Answers to Self-Test Exercises 809

Programming Projects 818

## Chapter 18 Exception Handling 825

### 18.1 EXCEPTION HANDLING BASICS 827

- A Toy Example of Exception Handling 827
- Defining Your Own Exception Classes 836
- Multiple Throws and Catches 836
- PITFALL: Catch the More Specific Exception First 840
- TIP: Exception Classes Can Be Trivial 841
- Throwing an Exception in a Function 841
- Exception Specification 843
- PITFALL: Exception Specification in Derived Classes 845

### 18.2 PROGRAMMING TECHNIQUES FOR EXCEPTION HANDLING 846

- When to Throw an Exception 847
- PITFALL: Uncaught Exceptions 848
- PITFALL: Nested `try-catch` Blocks 849
- PITFALL: Overuse of Exceptions 849
- Exception Class Hierarchies 850
- Testing for Available Memory 850
- Rethrowing an Exception 851
  
- Chapter Summary 851
- Answers to Self-Test Exercises 851
- Programming Projects 853

## Chapter 19 Standard Template Library 857

### 19.1 ITERATORS 859

- Iterator Basics 859
- PITFALL: Compiler Problems 864
- Kinds of Iterators 865
- Constant and Mutable Iterators 868
- Reverse Iterators 870
- Other Kinds of Iterators 871

### 19.2 CONTAINERS 872

- Sequential Containers 872
- PITFALL: Iterators and Removing Elements 877
- TIP: Type Definitions in Containers 878
- The Container Adapters `stack` and `queue` 878
- PITFALL: Underlying Containers 879
- The Associative Containers `set` and `map` 882
- Efficiency 887

**19.3 GENERIC ALGORITHMS 889**

- Running Times and Big-O Notation 890
- Container Access Running Times 894
- Nonmodifying Sequence Algorithms 895
- Modifying Sequence Algorithms 899
- Set Algorithms 900
- Sorting Algorithms 902
  
- Chapter Summary 902
- Answers to Self-Test Exercises 903
- Programming Projects 905

**Chapter 20 Patterns and UML (online at [www.pearsonhighered.com/savitch](http://www.pearsonhighered.com/savitch))**

- Appendix 1 C++ Keywords 915**
- Appendix 2 Precedence of Operators 917**
- Appendix 3 The ASCII Character Set 919**
- Appendix 4 Some Library Functions 921**
- Appendix 5 Old and New Header Files 929**
- Index 931**