



# Contents

<b>Preface</b>	<b>21</b>
<b>I Introduction to Computers and C++</b>	<b>35</b>
1.1 Introduction	36
1.2 Computers and the Internet in Industry and Research	36
1.3 Hardware and Software	39
1.3.1 Moore's Law	40
1.3.2 Computer Organization	40
1.4 Data Hierarchy	41
1.5 Machine Languages, Assembly Languages and High-Level Languages	43
1.6 C++	44
1.7 Programming Languages	45
1.8 Introduction to Object Technology	48
1.9 Typical C++ Development Environment	51
1.10 Test-Driving a C++ Application	53
1.11 Operating Systems	59
1.11.1 Windows—A Proprietary Operating System	59
1.11.2 Linux—An Open-Source Operating System	60
1.11.3 Apple's OS X; Apple's iOS for iPhone <sup>®</sup> , iPad <sup>®</sup> and iPod Touch <sup>®</sup> Devices	60
1.11.4 Google's Android	61
1.12 The Internet and World Wide Web	61
1.13 Some Key Software Development Terminology	63
1.14 C++11 and the Open Source Boost Libraries	65
1.15 Keeping Up to Date with Information Technologies	66
1.16 Web Resources	67
<b>2 Introduction to C++ Programming; Input/Output and Operators</b>	<b>72</b>
2.1 Introduction	73
2.2 First Program in C++: Printing a Line of Text	73

2.3	Modifying Our First C++ Program	77
2.4	Another C++ Program: Adding Integers	78
2.5	Memory Concepts	82
2.6	Arithmetic	83
2.7	Decision Making: Equality and Relational Operators	87
2.8	Wrap-Up	91
<b>3</b>	<b>Introduction to Classes, Objects and Strings</b>	<b>100</b>
3.1	Introduction	101
3.2	Defining a Class with a Member Function	101
3.3	Defining a Member Function with a Parameter	104
3.4	Data Members, <i>set</i> Member Functions and <i>get</i> Member Functions	108
3.5	Initializing Objects with Constructors	113
3.6	Placing a Class in a Separate File for Reusability	117
3.7	Separating Interface from Implementation	121
3.8	Validating Data with <i>set</i> Functions	126
3.9	Wrap-Up	131
<b>4</b>	<b>Control Statements: Part I; Assignment, ++ and -- Operators</b>	<b>138</b>
4.1	Introduction	139
4.2	Algorithms	139
4.3	Pseudocode	140
4.4	Control Structures	141
4.5	<i>if</i> Selection Statement	144
4.6	<i>if...else</i> Double-Selection Statement	146
4.7	<i>while</i> Repetition Statement	150
4.8	Formulating Algorithms: Counter-Controlled Repetition	152
4.9	Formulating Algorithms: Sentinel-Controlled Repetition	158
4.10	Formulating Algorithms: Nested Control Statements	168
4.11	Assignment Operators	173
4.12	Increment and Decrement Operators	174
4.13	Wrap-Up	177
<b>5</b>	<b>Control Statements: Part 2; Logical Operators</b>	<b>191</b>
5.1	Introduction	192
5.2	Essentials of Counter-Controlled Repetition	192
5.3	<i>for</i> Repetition Statement	193

5.4	Examples Using the for Statement	197
5.5	do...while Repetition Statement	202
5.6	switch Multiple-Selection Statement	203
5.7	break and continue Statements	212
5.8	Logical Operators	214
5.9	Confusing the Equality (==) and Assignment (=) Operators	219
5.10	Structured Programming Summary	220
5.11	Wrap-Up	225

## **6 Functions and an Introduction to Recursion 235**

6.1	Introduction	236
6.2	Program Components in C++	237
6.3	Math Library Functions	238
6.4	Function Definitions with Multiple Parameters	239
6.5	Function Prototypes and Argument Coercion	244
6.6	C++ Standard Library Headers	246
6.7	Case Study: Random Number Generation	248
6.8	Case Study: Game of Chance; Introducing enum	253
6.9	C++11 Random Numbers	258
6.10	Storage Classes and Storage Duration	259
6.11	Scope Rules	262
6.12	Function Call Stack and Activation Records	265
6.13	Functions with Empty Parameter Lists	269
6.14	Inline Functions	270
6.15	References and Reference Parameters	271
6.16	Default Arguments	274
6.17	Unary Scope Resolution Operator	276
6.18	Function Overloading	277
6.19	Function Templates	280
6.20	Recursion	282
6.21	Example Using Recursion: Fibonacci Series	286
6.22	Recursion vs. Iteration	289
6.23	Wrap-Up	292

## **7 Class Templates array and vector; Catching Exceptions 312**

7.1	Introduction	313
7.2	arrays	313
7.3	Declaring arrays	315

7.4	Examples Using arrays	315
7.4.1	Declaring an array and Using a Loop to Initialize the array's Elements	315
7.4.2	Initializing an array in a Declaration with an Initializer List	316
7.4.3	Specifying an array's Size with a Constant Variable and Setting array Elements with Calculations	317
7.4.4	Summing the Elements of an array	320
7.4.5	Using Bar Charts to Display array Data Graphically	320
7.4.6	Using the Elements of an array as Counters	322
7.4.7	Using arrays to Summarize Survey Results	323
7.4.8	Static Local arrays and Automatic Local arrays	325
7.5	Range-Based for Statement	327
7.6	Case Study: Class <code>GradeBook</code> Using an array to Store Grades	329
7.7	Sorting and Searching arrays	336
7.8	Multidimensional arrays	338
7.9	Case Study: Class <code>GradeBook</code> Using a Two-Dimensional array	341
7.10	Introduction to C++ Standard Library Class Template <code>vector</code>	348
7.11	Wrap-Up	354
<b>8</b>	<b>Pointers</b>	<b>368</b>
8.1	Introduction	369
8.2	Pointer Variable Declarations and Initialization	369
8.3	Pointer Operators	371
8.4	Pass-by-Reference with Pointers	373
8.5	Built-In Arrays	378
8.6	Using <code>const</code> with Pointers	380
8.6.1	Nonconstant Pointer to Nonconstant Data	381
8.6.2	Nonconstant Pointer to Constant Data	381
8.6.3	Constant Pointer to Nonconstant Data	382
8.6.4	Constant Pointer to Constant Data	383
8.7	<code>sizeof</code> Operator	384
8.8	Pointer Expressions and Pointer Arithmetic	387
8.9	Relationship Between Pointers and Built-In Arrays	389
8.10	Pointer-Based Strings	392
8.11	Wrap-Up	395
<b>9</b>	<b>Classes: A Deeper Look; Throwing Exceptions</b>	<b>411</b>
9.1	Introduction	412
9.2	Time Class Case Study	413

9.3	Class Scope and Accessing Class Members	419
9.4	Access Functions and Utility Functions	420
9.5	Time Class Case Study: Constructors with Default Arguments	421
9.6	Destructors	427
9.7	When Constructors and Destructors Are Called	427
9.8	Time Class Case Study: A Subtle Trap—Returning a Reference or a Pointer to a <code>private</code> Data Member	431
9.9	Default Memberwise Assignment	434
9.10	<code>const</code> Objects and <code>const</code> Member Functions	436
9.11	Composition: Objects as Members of Classes	438
9.12	<code>friend</code> Functions and <code>friend</code> Classes	444
9.13	Using the <code>this</code> Pointer	446
9.14	<code>static</code> Class Members	452
9.15	Wrap-Up	457

## **10 Operator Overloading; Class `string` 467**

10.1	Introduction	468
10.2	Using the Overloaded Operators of Standard Library Class <code>string</code>	469
10.3	Fundamentals of Operator Overloading	472
10.4	Overloading Binary Operators	473
10.5	Overloading the Binary Stream Insertion and Stream Extraction Operators	474
10.6	Overloading Unary Operators	478
10.7	Overloading the Unary Prefix and Postfix <code>++</code> and <code>--</code> Operators	479
10.8	Case Study: A Date Class	480
10.9	Dynamic Memory Management	485
10.10	Case Study: Array Class	487
	10.10.1 Using the Array Class	488
	10.10.2 Array Class Definition	492
10.11	Operators as Member vs. Non-Member Functions	500
10.12	Converting Between Types	500
10.13	<code>explicit</code> Constructors and Conversion Operators	502
10.14	Overloading the Function Call Operator <code>()</code>	504
10.15	Wrap-Up	505

## **11 Object-Oriented Programming: Inheritance 516**

11.1	Introduction	517
11.2	Base Classes and Derived Classes	517

11.3	Relationship between Base and Derived Classes	520
11.3.1	Creating and Using a <code>CommissionEmployee</code> Class	520
11.3.2	Creating a <code>BasePlusCommissionEmployee</code> Class Without Using Inheritance	525
11.3.3	Creating a <code>CommissionEmployee</code> – <code>BasePlusCommissionEmployee</code> Inheritance Hierarchy	531
11.3.4	<code>CommissionEmployee</code> – <code>BasePlusCommissionEmployee</code> Inheritance Hierarchy Using protected Data	535
11.3.5	<code>CommissionEmployee</code> – <code>BasePlusCommissionEmployee</code> Inheritance Hierarchy Using private Data	538
11.4	Constructors and Destructors in Derived Classes	543
11.5	<code>public</code> , <code>protected</code> and <code>private</code> Inheritance	545
11.6	Software Engineering with Inheritance	546
11.7	Wrap-Up	546

## **12 Object-Oriented Programming: Polymorphism 551**

12.1	Introduction	552
12.2	Introduction to Polymorphism: Polymorphic Video Game	553
12.3	Relationships Among Objects in an Inheritance Hierarchy	553
12.3.1	Invoking Base-Class Functions from Derived-Class Objects	554
12.3.2	Aiming Derived-Class Pointers at Base-Class Objects	557
12.3.3	Derived-Class Member-Function Calls via Base-Class Pointers	558
12.3.4	Virtual Functions and Virtual Destructors	560
12.4	Type Fields and <code>switch</code> Statements	567
12.5	Abstract Classes and Pure virtual Functions	567
12.6	Case Study: Payroll System Using Polymorphism	569
12.6.1	Creating Abstract Base Class <code>Employee</code>	570
12.6.2	Creating Concrete Derived Class <code>SalariedEmployee</code>	574
12.6.3	Creating Concrete Derived Class <code>CommissionEmployee</code>	576
12.6.4	Creating Indirect Concrete Derived Class <code>BasePlusCommissionEmployee</code>	578
12.6.5	Demonstrating Polymorphic Processing	580
12.7	(Optional) Polymorphism, Virtual Functions and Dynamic Binding “Under the Hood”	584
12.8	Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, <code>dynamic_cast</code> , <code>typeid</code> and <code>type_info</code>	587
12.9	Wrap-Up	591

<b>13</b>	<b>Stream Input/Output: A Deeper Look</b>	<b>596</b>
13.1	Introduction	597
13.2	Streams	598
	13.2.1 Classic Streams vs. Standard Streams	598
	13.2.2 <code>iostream</code> Library Headers	599
	13.2.3 Stream Input/Output Classes and Objects	599
13.3	Stream Output	601
	13.3.1 Output of <code>char *</code> Variables	602
	13.3.2 Character Output Using Member Function <code>put</code>	602
13.4	Stream Input	603
	13.4.1 <code>get</code> and <code>getline</code> Member Functions	603
	13.4.2 <code>istream</code> Member Functions <code>peek</code> , <code>putback</code> and <code>ignore</code>	606
	13.4.3 Type-Safe I/O	606
13.5	Unformatted I/O Using <code>read</code> , <code>write</code> and <code>gcount</code>	606
13.6	Introduction to Stream Manipulators	607
	13.6.1 Integral Stream Base: <code>dec</code> , <code>oct</code> , <code>hex</code> and <code>setbase</code>	608
	13.6.2 Floating-Point Precision ( <code>precision</code> , <code>setprecision</code> )	608
	13.6.3 Field Width ( <code>width</code> , <code>setw</code> )	610
	13.6.4 User-Defined Output Stream Manipulators	611
13.7	Stream Format States and Stream Manipulators	612
	13.7.1 Trailing Zeros and Decimal Points ( <code>showpoint</code> )	613
	13.7.2 Justification ( <code>left</code> , <code>right</code> and <code>internal</code> )	614
	13.7.3 Padding ( <code>fill</code> , <code>setfill</code> )	616
	13.7.4 Integral Stream Base ( <code>dec</code> , <code>oct</code> , <code>hex</code> , <code>showbase</code> )	617
	13.7.5 Floating-Point Numbers; Scientific and Fixed Notation ( <code>scientific</code> , <code>fixed</code> )	618
	13.7.6 Uppercase/Lowercase Control ( <code>uppercase</code> )	619
	13.7.7 Specifying Boolean Format ( <code>boolalpha</code> )	619
	13.7.8 Setting and Resetting the Format State via Member Function <code>flags</code>	620
13.8	Stream Error States	621
13.9	Tying an Output Stream to an Input Stream	624
13.10	Wrap-Up	624
<b>14</b>	<b>File Processing</b>	<b>633</b>
14.1	Introduction	634
14.2	Files and Streams	634
14.3	Creating a Sequential File	635
14.4	Reading Data from a Sequential File	639
14.5	Updating Sequential Files	645

14.6	Random-Access Files	645
14.7	Creating a Random-Access File	646
14.8	Writing Data Randomly to a Random-Access File	651
14.9	Reading from a Random-Access File Sequentially	653
14.10	Case Study: A Transaction-Processing Program	655
14.11	Object Serialization	662
14.12	Wrap-Up	662
<b>15</b>	<b>Standard Library Containers and Iterators</b>	<b>672</b>
15.1	Introduction	673
15.2	Introduction to Containers	674
15.3	Introduction to Iterators	678
15.4	Introduction to Algorithms	683
15.5	Sequence Containers	683
15.5.1	vector Sequence Container	684
15.5.2	list Sequence Container	692
15.5.3	deque Sequence Container	696
15.6	Associative Containers	698
15.6.1	multiset Associative Container	699
15.6.2	set Associative Container	702
15.6.3	multimap Associative Container	703
15.6.4	map Associative Container	705
15.7	Container Adapters	707
15.7.1	stack Adapter	707
15.7.2	queue Adapter	709
15.7.3	priority_queue Adapter	710
15.8	Class bitset	711
15.9	Wrap-Up	713
<b>16</b>	<b>Standard Library Algorithms</b>	<b>724</b>
16.1	Introduction	725
16.2	Minimum Iterator Requirements	725
16.3	Algorithms	727
16.3.1	fill, fill_n, generate and generate_n	727
16.3.2	equal, mismatch and lexicographical_compare	729
16.3.3	remove, remove_if, remove_copy and remove_copy_if	731
16.3.4	replace, replace_if, replace_copy and replace_copy_if	734
16.3.5	Mathematical Algorithms	736
16.3.6	Basic Searching and Sorting Algorithms	740



16.3.7	swap, iter_swap and swap_ranges	744
16.3.8	copy_backward, merge, unique and reverse	745
16.3.9	inplace_merge, unique_copy and reverse_copy	748
16.3.10	Set Operations	750
16.3.11	lower_bound, upper_bound and equal_range	753
16.3.12	Heapsort	755
16.3.13	min, max, minmax and minmax_element	758
16.4	Function Objects	760
16.5	Lambda Expressions	763
16.6	Standard Library Algorithm Summary	764
16.7	Wrap-Up	766

## **17 Exception Handling: A Deeper Look 774**

17.1	Introduction	775
17.2	Example: Handling an Attempt to Divide by Zero	775
17.3	Rethrowing an Exception	781
17.4	Stack Unwinding	782
17.5	When to Use Exception Handling	784
17.6	Constructors, Destructors and Exception Handling	785
17.7	Exceptions and Inheritance	786
17.8	Processing new Failures	786
17.9	Class unique_ptr and Dynamic Memory Allocation	789
17.10	Standard Library Exception Hierarchy	792
17.11	Wrap-Up	793

## **18 Introduction to Custom Templates 799**

18.1	Introduction	800
18.2	Class Templates	800
18.3	Function Template to Manipulate a Class-Template Specialization Object	805
18.4	Nontype Parameters	807
18.5	Default Arguments for Template Type Parameters	807
18.6	Overloading Function Templates	808
18.7	Wrap-Up	808

## **19 Custom Templated Data Structures 811**

19.1	Introduction	812
19.2	Self-Referential Classes	813
19.3	Linked Lists	814

19.4	Stacks	828
19.5	Queues	833
19.6	Trees	837
19.7	Wrap-Up	845
<b>20</b>	<b>Searching and Sorting</b>	<b>856</b>
20.1	Introduction	857
20.2	Searching Algorithms	858
20.2.1	Linear Search	858
20.2.2	Binary Search	861
20.3	Sorting Algorithms	865
20.3.1	Insertion Sort	866
20.3.2	Selection Sort	868
20.3.3	Merge Sort (A Recursive Implementation)	871
20.4	Wrap-Up	877
<b>21</b>	<b>Class string and String Stream Processing: A Deeper Look</b>	<b>883</b>
21.1	Introduction	884
21.2	string Assignment and Concatenation	885
21.3	Comparing strings	887
21.4	Substrings	890
21.5	Swapping strings	890
21.6	string Characteristics	891
21.7	Finding Substrings and Characters in a string	893
21.8	Replacing Characters in a string	895
21.9	Inserting Characters into a string	897
21.10	Conversion to Pointer-Based char * Strings	898
21.11	Iterators	899
21.12	String Stream Processing	901
21.13	C++11 Numeric Conversion Functions	904
21.14	Wrap-Up	905
<b>22</b>	<b>Bits, Characters, C Strings and structs</b>	<b>913</b>
22.1	Introduction	914
22.2	Structure Definitions	914
22.3	typedef	916
22.4	Example: Card Shuffling and Dealing Simulation	916

22.5	Bitwise Operators	919
22.6	Bit Fields	928
22.7	Character-Handling Library	931
22.8	C String-Manipulation Functions	937
22.9	C String-Conversion Functions	944
22.10	Search Functions of the C String-Handling Library	949
22.11	Memory Functions of the C String-Handling Library	953
22.12	Wrap-Up	957

## **23 Other Topics 972**

23.1	Introduction	973
23.2	const_cast Operator	973
23.3	mutable Class Members	975
23.4	namespaces	977
23.5	Operator Keywords	980
23.6	Pointers to Class Members (. * and ->*)	982
23.7	Multiple Inheritance	984
23.8	Multiple Inheritance and virtual Base Classes	989
23.9	Wrap-Up	993

## **List of Chapters on the Web 999**

### **A Operator Precedence and Associativity 1001**

### **B ASCII Character Set 1003**

### **C Fundamental Types 1004**

### **D Number Systems 1006**

D.1	Introduction	1007
D.2	Abbreviating Binary Numbers as Octal and Hexadecimal Numbers	1010
D.3	Converting Octal and Hexadecimal Numbers to Binary Numbers	1011
D.4	Converting from Binary, Octal or Hexadecimal to Decimal	1011
D.5	Converting from Decimal to Binary, Octal or Hexadecimal	1012
D.6	Negative Binary Numbers: Two's Complement Notation	1014

<b>E</b>	<b>Preprocessor</b>	<b>1019</b>
E.1	Introduction	1020
E.2	#include Preprocessing Directive	1020
E.3	#define Preprocessing Directive: Symbolic Constants	1021
E.4	#define Preprocessing Directive: Macros	1021
E.5	Conditional Compilation	1023
E.6	#error and #pragma Preprocessing Directives	1024
E.7	Operators # and ##	1025
E.8	Predefined Symbolic Constants	1025
E.9	Assertions	1026
E.10	Wrap-Up	1026

## List of Appendices on the Web **1031**

## Index **1033**

## Online Chapters and Appendices

<b>24</b>	<b>C++11 Additional Features</b>	<b>24-1</b>
<b>25</b>	<b>ATM Case Study, Part 1: Object-Oriented Design with the UML</b>	<b>25-1</b>
25.1	Introduction	25-2
25.2	Introduction to Object-Oriented Analysis and Design	25-2
25.3	Examining the ATM Requirements Document	25-3
25.4	Identifying the Classes in the ATM Requirements Document	25-10
25.5	Identifying Class Attributes	25-17
25.6	Identifying Objects' States and Activities	25-21
25.7	Identifying Class Operations	25-25
25.8	Indicating Collaboration Among Objects	25-32
25.9	Wrap-Up	25-39
<b>26</b>	<b>ATM Case Study, Part 2: Implementing an Object-Oriented Design</b>	<b>26-1</b>
26.1	Introduction	26-2
26.2	Starting to Program the Classes of the ATM System	26-2

26.3	Incorporating Inheritance into the ATM System	26-8
26.4	ATM Case Study Implementation	26-15
26.4.1	Class ATM	26-16
26.4.2	Class Screen	26-23
26.4.3	Class Keypad	26-25
26.4.4	Class CashDispenser	26-26
26.4.5	Class DepositSlot	26-28
26.4.6	Class Account	26-29
26.4.7	Class BankDatabase	26-31
26.4.8	Class Transaction	26-35
26.4.9	Class BalanceInquiry	26-37
26.4.10	Class Withdrawal	26-39
26.4.11	Class Deposit	26-44
26.4.12	Test Program ATMCaseStudy.cpp	26-47
26.5	Wrap-Up	26-47

## **F C Legacy Code Topics F-1**

F.1	Introduction	F-2
F.2	Redirecting Input/Output on UNIX/Linux/Mac OS X and Windows Systems	F-2
F.3	Variable-Length Argument Lists	F-3
F.4	Using Command-Line Arguments	F-5
F.5	Notes on Compiling Multiple-Source-File Programs	F-7
F.6	Program Termination with <code>exit</code> and <code>atexit</code>	F-9
F.7	Type Qualifier <code>volatile</code>	F-10
F.8	Suffixes for Integer and Floating-Point Constants	F-10
F.9	Signal Handling	F-11
F.10	Dynamic Memory Allocation with <code>calloc</code> and <code>realloc</code>	F-13
F.11	Unconditional Branch: <code>goto</code>	F-14
F.12	Unions	F-15
F.13	Linkage Specifications	F-18
F.14	Wrap-Up	F-19

## **G UML 2: Additional Diagram Types G-1**

G.1	Introduction	G-1
G.2	Additional Diagram Types	G-2

<b>H</b>	<b>Using the Visual Studio Debugger</b>	<b>H-1</b>
H.1	Introduction	H-2
H.2	Breakpoints and the Continue Command	H-2
H.3	Locals and Watch Windows	H-8
H.4	Controlling Execution Using the Step Into, Step Over, Step Out and Continue Commands	H-11
H.5	Autos Window	H-13
H.6	Wrap-Up	H-14
<b>I</b>	<b>Using the GNU C++ Debugger</b>	<b>I-1</b>
I.1	Introduction	I-2
I.2	Breakpoints and the run, stop, continue and print Commands	I-2
I.3	print and set Commands	I-8
I.4	Controlling Execution Using the step, finish and next Commands	I-10
I.5	watch Command	I-13
I.6	Wrap-Up	I-15
<b>J</b>	<b>Using the Xcode Debugger</b>	<b>J-1</b>
<b>K</b>	<b>Test Driving a C++ Program on Mac OS X</b>	<b>K-1</b>
	[Note: The test drives for Windows and Linux are in Chapter 1.]	