

Contents

Chapter 24 and Appendices F–K are PDF documents posted online at
www.informit.com/title/9780133439854

Preface	xix
1 Introduction	1
1.1 Introduction	2
1.2 C++	2
1.3 Object Technology	3
1.4 Typical C++ Development Environment	6
1.5 Test-Driving a C++ Application	9
1.6 Operating Systems	15
1.6.1 Windows—A Proprietary Operating System	15
1.6.2 Linux—An Open-Source Operating System	15
1.6.3 Apple’s OS X; Apple’s iOS for iPhone®, iPad® and iPod Touch® Devices	16
1.6.4 Google’s Android	16
1.7 C++11 and the Open Source Boost Libraries	17
1.8 Web Resources	18
2 Introduction to C++ Programming, Input/Output and Operators	19
2.1 Introduction	20
2.2 First Program in C++: Printing a Line of Text	20
2.3 Modifying Our First C++ Program	23
2.4 Another C++ Program: Adding Integers	24
2.5 Arithmetic	28
2.6 Decision Making: Equality and Relational Operators	32
2.7 Wrap-Up	35
3 Introduction to Classes, Objects and Strings	37
3.1 Introduction	38
3.2 Defining a Class with a Member Function	38
3.3 Defining a Member Function with a Parameter	41
3.4 Data Members, <i>set</i> Member Functions and <i>get</i> Member Functions	44
3.5 Initializing Objects with Constructors	50

viii Contents

3.6	Placing a Class in a Separate File for Reusability	54
3.7	Separating Interface from Implementation	58
3.8	Validating Data with <i>set</i> Functions	63
3.9	Wrap-Up	68
4	Control Statements: Part 1; Assignment, ++ and -- Operators	69
4.1	Introduction	70
4.2	Control Structures	70
4.3	<i>if</i> Selection Statement	73
4.4	<i>if...else</i> Double-Selection Statement	74
4.5	<i>while</i> Repetition Statement	78
4.6	Counter-Controlled Repetition	79
4.7	Sentinel-Controlled Repetition	85
4.8	Nested Control Statements	92
4.9	Assignment Operators	95
4.10	Increment and Decrement Operators	96
4.11	Wrap-Up	99
5	Control Statements: Part 2; Logical Operators	100
5.1	Introduction	101
5.2	Essentials of Counter-Controlled Repetition	101
5.3	<i>for</i> Repetition Statement	102
5.4	Examples Using the <i>for</i> Statement	106
5.5	<i>do...while</i> Repetition Statement	110
5.6	<i>switch</i> Multiple-Selection Statement	112
5.7	<i>break</i> and <i>continue</i> Statements	121
5.8	Logical Operators	122
5.9	Confusing the Equality (==) and Assignment (=) Operators	127
5.10	Wrap-Up	128
6	Functions and an Introduction to Recursion	129
6.1	Introduction	130
6.2	Math Library Functions	130
6.3	Function Definitions with Multiple Parameters	132
6.4	Function Prototypes and Argument Coercion	137
6.5	C++ Standard Library Headers	139
6.6	Case Study: Random Number Generation	141
6.7	Case Study: Game of Chance; Introducing <i>enum</i>	146
6.8	C++11 Random Numbers	151
6.9	Storage Classes and Storage Duration	152
6.10	Scope Rules	155
6.11	Function Call Stack and Activation Records	158

6.12	Functions with Empty Parameter Lists	162
6.13	Inline Functions	163
6.14	References and Reference Parameters	164
6.15	Default Arguments	167
6.16	Unary Scope Resolution Operator	169
6.17	Function Overloading	170
6.18	Function Templates	173
6.19	Recursion	175
6.20	Example Using Recursion: Fibonacci Series	179
6.21	Recursion vs. Iteration	182
6.22	Wrap-Up	184

7 Class Templates array and vector; Catching Exceptions **185**

7.1	Introduction	186
7.2	arrays	186
7.3	Declaring arrays	188
7.4	Examples Using arrays	188
7.4.1	Declaring an array and Using a Loop to Initialize the array's Elements	188
7.4.2	Initializing an array in a Declaration with an Initializer List	189
7.4.3	Specifying an array's Size with a Constant Variable and Setting array Elements with Calculations	190
7.4.4	Summing the Elements of an array	192
7.4.5	Using Bar Charts to Display array Data Graphically	193
7.4.6	Using the Elements of an array as Counters	195
7.4.7	Using arrays to Summarize Survey Results	196
7.4.8	Static Local arrays and Automatic Local arrays	198
7.5	Range-Based for Statement	200
7.6	Case Study: Class GradeBook Using an array to Store Grades	202
7.7	Sorting and Searching arrays	209
7.8	Multidimensional arrays	211
7.9	Case Study: Class GradeBook Using a Two-Dimensional array	214
7.10	Introduction to C++ Standard Library Class Template vector	221
7.11	Wrap-Up	227

8 Pointers **228**

8.1	Introduction	229
8.2	Pointer Variable Declarations and Initialization	229
8.3	Pointer Operators	231
8.4	Pass-by-Reference with Pointers	233
8.5	Built-In Arrays	238
8.6	Using const with Pointers	240
8.6.1	Nonconstant Pointer to Nonconstant Data	241
8.6.2	Nonconstant Pointer to Constant Data	241

x Contents

8.6.3	Constant Pointer to Nonconstant Data	243
8.6.4	Constant Pointer to Constant Data	243
8.7	sizeof Operator	244
8.8	Pointer Expressions and Pointer Arithmetic	247
8.9	Relationship Between Pointers and Built-In Arrays	249
8.10	Pointer-Based Strings	252
8.11	Wrap-Up	255

9 Classes: A Deeper Look; Throwing Exceptions 256

9.1	Introduction	257
9.2	Time Class Case Study	258
9.3	Class Scope and Accessing Class Members	264
9.4	Access Functions and Utility Functions	265
9.5	Time Class Case Study: Constructors with Default Arguments	266
9.6	Destructors	272
9.7	When Constructors and Destructors Are Called	272
9.8	Time Class Case Study: A Subtle Trap—Returning a Reference or a Pointer to a private Data Member	276
9.9	Default Memberwise Assignment	279
9.10	const Objects and const Member Functions	281
9.11	Composition: Objects as Members of Classes	283
9.12	friend Functions and friend Classes	289
9.13	Using the this Pointer	291
9.14	static Class Members	297
9.15	Wrap-Up	302

10 Operator Overloading; Class string 303

10.1	Introduction	304
10.2	Using the Overloaded Operators of Standard Library Class string	305
10.3	Fundamentals of Operator Overloading	308
10.4	Overloading Binary Operators	309
10.5	Overloading the Binary Stream Insertion and Stream Extraction Operators	310
10.6	Overloading Unary Operators	314
10.7	Overloading the Unary Prefix and Postfix ++ and -- Operators	315
10.8	Case Study: A Date Class	316
10.9	Dynamic Memory Management	321
10.10	Case Study: Array Class	323
	10.10.1 Using the Array Class	324
	10.10.2 Array Class Definition	328
10.11	Operators as Member vs. Non-Member Functions	336
10.12	Converting Between Types	337
10.13	explicit Constructors and Conversion Operators	338
10.14	Overloading the Function Call Operator ()	340
10.15	Wrap-Up	341

11	Object-Oriented Programming: Inheritance	342
11.1	Introduction	343
11.2	Base Classes and Derived Classes	343
11.3	Relationship between Base and Derived Classes	346
11.3.1	Creating and Using a <code>CommissionEmployee</code> Class	346
11.3.2	Creating a <code>BasePlusCommissionEmployee</code> Class Without Using Inheritance	351
11.3.3	Creating a <code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy	357
11.3.4	<code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy Using <code>protected</code> Data	361
11.3.5	<code>CommissionEmployee–BasePlusCommissionEmployee</code> Inheritance Hierarchy Using <code>private</code> Data	364
11.4	Constructors and Destructors in Derived Classes	369
11.5	<code>public</code> , <code>protected</code> and <code>private</code> Inheritance	371
11.6	Software Engineering with Inheritance	372
11.7	Wrap-Up	372
12	Object-Oriented Programming: Polymorphism	374
12.1	Introduction	375
12.2	Introduction to Polymorphism: Polymorphic Video Game	376
12.3	Relationships Among Objects in an Inheritance Hierarchy	376
12.3.1	Invoking Base-Class Functions from Derived-Class Objects	377
12.3.2	Aiming Derived-Class Pointers at Base-Class Objects	380
12.3.3	Derived-Class Member-Function Calls via Base-Class Pointers	381
12.3.4	Virtual Functions and Virtual Destructors	383
12.4	Type Fields and <code>switch</code> Statements	390
12.5	Abstract Classes and Pure <code>virtual</code> Functions	390
12.6	Case Study: Payroll System Using Polymorphism	392
12.6.1	Creating Abstract Base Class <code>Employee</code>	393
12.6.2	Creating Concrete Derived Class <code>SalariedEmployee</code>	397
12.6.3	Creating Concrete Derived Class <code>CommissionEmployee</code>	399
12.6.4	Creating Indirect Concrete Derived Class <code>BasePlusCommissionEmployee</code>	401
12.6.5	Demonstrating Polymorphic Processing	403
12.7	(Optional) Polymorphism, Virtual Functions and Dynamic Binding “Under the Hood”	407
12.8	Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, <code>dynamic_cast</code> , <code>typeid</code> and <code>typeid</code>	410
12.9	Wrap-Up	414
13	Stream Input/Output: A Deeper Look	415
13.1	Introduction	416

xii Contents

13.2	Streams	417
13.2.1	Classic Streams vs. Standard Streams	417
13.2.2	<code>iostream</code> Library Headers	418
13.2.3	Stream Input/Output Classes and Objects	418
13.3	Stream Output	420
13.3.1	Output of <code>char *</code> Variables	421
13.3.2	Character Output Using Member Function <code>put</code>	421
13.4	Stream Input	422
13.4.1	<code>get</code> and <code>getline</code> Member Functions	422
13.4.2	<code>istream</code> Member Functions <code>peek</code> , <code>putback</code> and <code>ignore</code>	425
13.4.3	Type-Safe I/O	425
13.5	Unformatted I/O Using <code>read</code> , <code>write</code> and <code>gcount</code>	425
13.6	Introduction to Stream Manipulators	426
13.6.1	Integral Stream Base: <code>dec</code> , <code>oct</code> , <code>hex</code> and <code>setbase</code>	427
13.6.2	Floating-Point Precision (<code>precision</code> , <code>setprecision</code>)	427
13.6.3	Field Width (<code>width</code> , <code>setw</code>)	429
13.6.4	User-Defined Output Stream Manipulators	430
13.7	Stream Format States and Stream Manipulators	431
13.7.1	Trailing Zeros and Decimal Points (<code>showpoint</code>)	432
13.7.2	Justification (<code>left</code> , <code>right</code> and <code>internal</code>)	433
13.7.3	Padding (<code>fill</code> , <code>setfill</code>)	435
13.7.4	Integral Stream Base (<code>dec</code> , <code>oct</code> , <code>hex</code> , <code>showbase</code>)	436
13.7.5	Floating-Point Numbers; Scientific and Fixed Notation (<code>scientific</code> , <code>fixed</code>)	437
13.7.6	Uppercase/Lowercase Control (<code>uppercase</code>)	438
13.7.7	Specifying Boolean Format (<code>boolalpha</code>)	438
13.7.8	Setting and Resetting the Format State via Member Function <code>flags</code>	439
13.8	Stream Error States	440
13.9	Tying an Output Stream to an Input Stream	443
13.10	Wrap-Up	443

14 File Processing 444

14.1	Introduction	445
14.2	Files and Streams	445
14.3	Creating a Sequential File	446
14.4	Reading Data from a Sequential File	450
14.5	Updating Sequential Files	456
14.6	Random-Access Files	456
14.7	Creating a Random-Access File	457
14.8	Writing Data Randomly to a Random-Access File	462
14.9	Reading from a Random-Access File Sequentially	464
14.10	Case Study: A Transaction-Processing Program	466
14.11	Object Serialization	473
14.12	Wrap-Up	473

15	Standard Library Containers and Iterators	474
15.1	Introduction	475
15.2	Introduction to Containers	476
15.3	Introduction to Iterators	480
15.4	Introduction to Algorithms	485
15.5	Sequence Containers	485
15.5.1	vector Sequence Container	486
15.5.2	list Sequence Container	494
15.5.3	deque Sequence Container	498
15.6	Associative Containers	500
15.6.1	multiset Associative Container	501
15.6.2	set Associative Container	504
15.6.3	multimap Associative Container	505
15.6.4	map Associative Container	507
15.7	Container Adapters	509
15.7.1	stack Adapter	509
15.7.2	queue Adapter	511
15.7.3	priority_queue Adapter	512
15.8	Class bitset	513
15.9	Wrap-Up	515
16	Standard Library Algorithms	517
16.1	Introduction	518
16.2	Minimum Iterator Requirements	518
16.3	Algorithms	520
16.3.1	fill, fill_n, generate and generate_n	520
16.3.2	equal, mismatch and lexicographical_compare	522
16.3.3	remove, remove_if, remove_copy and remove_copy_if	524
16.3.4	replace, replace_if, replace_copy and replace_copy_if	527
16.3.5	Mathematical Algorithms	529
16.3.6	Basic Searching and Sorting Algorithms	533
16.3.7	swap, iter_swap and swap_ranges	537
16.3.8	copy_backward, merge, unique and reverse	538
16.3.9	inplace_merge, unique_copy and reverse_copy	541
16.3.10	Set Operations	543
16.3.11	lower_bound, upper_bound and equal_range	546
16.3.12	Heapsort	548
16.3.13	min, max, minmax and minmax_element	551
16.4	Function Objects	553
16.5	Lambda Expressions	556
16.6	Standard Library Algorithm Summary	557
16.7	Wrap-Up	559
17	Exception Handling: A Deeper Look	560
17.1	Introduction	561

xiv Contents

17.2	Example: Handling an Attempt to Divide by Zero	561
17.3	Rethrowing an Exception	567
17.4	Stack Unwinding	568
17.5	When to Use Exception Handling	570
17.6	Constructors, Destructors and Exception Handling	571
17.7	Exceptions and Inheritance	572
17.8	Processing new Failures	572
17.9	Class <code>unique_ptr</code> and Dynamic Memory Allocation	575
17.10	Standard Library Exception Hierarchy	578
17.11	Wrap-Up	579

18 Introduction to Custom Templates 581

18.1	Introduction	582
18.2	Class Templates	582
18.3	Function Template to Manipulate a Class-Template Specialization Object	587
18.4	Nontype Parameters	589
18.5	Default Arguments for Template Type Parameters	589
18.6	Overloading Function Templates	589
18.7	Wrap-Up	590

19 Class `string` and String Stream Processing: A Deeper Look 591

19.1	Introduction	592
19.2	<code>string</code> Assignment and Concatenation	593
19.3	Comparing strings	595
19.4	Substrings	598
19.5	Swapping strings	598
19.6	<code>string</code> Characteristics	599
19.7	Finding Substrings and Characters in a <code>string</code>	601
19.8	Replacing Characters in a <code>string</code>	603
19.9	Inserting Characters into a <code>string</code>	605
19.10	Conversion to Pointer-Based <code>char *</code> Strings	606
19.11	Iterators	607
19.12	String Stream Processing	609
19.13	C++11 Numeric Conversion Functions	612
19.14	Wrap-Up	613

20 Bits, Characters, C Strings and structs 615

20.1	Introduction	616
20.2	Structure Definitions	616
20.3	<code>typedef</code>	618
20.4	Example: Card Shuffling and Dealing Simulation	618
20.5	Bitwise Operators	621

20.6	Bit Fields	630
20.7	Character-Handling Library	633
20.8	C String-Manipulation Functions	639
20.9	C String-Conversion Functions	646
20.10	Search Functions of the C String-Handling Library	651
20.11	Memory Functions of the C String-Handling Library	655
20.12	Wrap-Up	659

21 Other Topics 660

21.1	Introduction	661
21.2	<code>const_cast</code> Operator	661
21.3	<code>mutable</code> Class Members	663
21.4	namespaces	665
21.5	Operator Keywords	668
21.6	Pointers to Class Members (<code>.</code> , <code>*</code> and <code>->*</code>)	670
21.7	Multiple Inheritance	672
21.8	Multiple Inheritance and <code>virtual</code> Base Classes	677
21.9	Wrap-Up	681

22 ATM Case Study, Part 1: Object-Oriented Design with the UML 682

22.1	Introduction	683
22.2	Introduction to Object-Oriented Analysis and Design	683
22.3	Examining the ATM Requirements Document	684
22.4	Identifying the Classes in the ATM Requirements Document	691
22.5	Identifying Class Attributes	698
22.6	Identifying Objects' States and Activities	703
22.7	Identifying Class Operations	707
22.8	Indicating Collaboration Among Objects	714
22.9	Wrap-Up	721

23 ATM Case Study, Part 2: Implementing an Object-Oriented Design 725

23.1	Introduction	726
23.2	Starting to Program the Classes of the ATM System	726
23.3	Incorporating Inheritance into the ATM System	732
23.4	ATM Case Study Implementation	739
23.4.1	Class <code>ATM</code>	740
23.4.2	Class <code>Screen</code>	747
23.4.3	Class <code>Keypad</code>	749
23.4.4	Class <code>CashDispenser</code>	750
23.4.5	Class <code>DepositSlot</code>	752

xvi Contents

23.4.6	Class Account	753
23.4.7	Class BankDatabase	755
23.4.8	Class Transaction	759
23.4.9	Class BalanceInquiry	761
23.4.10	Class Withdrawal	763
23.4.11	Class Deposit	768
23.4.12	Test Program ATMCaseStudy.cpp	771
23.5	Wrap-Up	771

A Operator Precedence and Associativity 774**B ASCII Character Set** 777**C Fundamental Types** 778**D Number Systems** 780

D.1	Introduction	781
D.2	Abbreviating Binary Numbers as Octal and Hexadecimal Numbers	784
D.3	Converting Octal and Hexadecimal Numbers to Binary Numbers	785
D.4	Converting from Binary, Octal or Hexadecimal to Decimal	785
D.5	Converting from Decimal to Binary, Octal or Hexadecimal	786
D.6	Negative Binary Numbers: Two's Complement Notation	788

E Preprocessor 790

E.1	Introduction	791
E.2	<code>#include</code> Preprocessing Directive	791
E.3	<code>#define</code> Preprocessing Directive: Symbolic Constants	792
E.4	<code>#define</code> Preprocessing Directive: Macros	792
E.5	Conditional Compilation	794
E.6	<code>#error</code> and <code>#pragma</code> Preprocessing Directives	795
E.7	Operators <code>#</code> and <code>##</code>	796
E.8	Predefined Symbolic Constants	796
E.9	Assertions	797
E.10	Wrap-Up	797

Index 799**Online Chapters and Appendices**

Chapter 24 and Appendices F–K are PDF documents posted online at www.informit.com/title/9780133439854

24 C++11 Additional Features 24-1

F	C Legacy Code Topics	F-1
G	UML 2: Additional Diagram Types	G-1
H	Using the Visual Studio Debugger	H-1
I	Using the GNU C++ Debugger	I-1
J	Using the Xcode Debugger	J-1
K	Test Driving a C++ Program on Mac OS X	K-1

[Note: The test drives for Windows and Linux are in Chapter 1.]

