



Contents

Chapters 25–26 and Appendices F–I are PDF documents posted online at the book’s Companion Website, which is accessible from www.pearsonhighered.com/deitel.

Preface

xxi

1	Introduction to Computers and C++	1
1.1	Introduction	2
1.2	Computers: Hardware and Software	5
1.3	Data Hierarchy	6
1.4	Computer Organization	7
1.5	Machine Languages, Assembly Languages and High-Level Languages	9
1.6	Introduction to Object Technology	10
1.7	Operating Systems	13
1.8	Programming Languages	15
1.9	C++ and a Typical C++ Development Environment	17
1.10	Test-Driving a C++ Application	21
1.11	Web 2.0: Going Social	27
1.12	Software Technologies	29
1.13	Future of C++: TR1, the New C++ Standard and the Open Source Boost Libraries	31
1.14	Keeping Up-to-Date with Information Technologies	32
1.15	Wrap-Up	32
2	Introduction to C++ Programming	37
2.1	Introduction	38
2.2	First Program in C++: Printing a Line of Text	38
2.3	Modifying Our First C++ Program	42
2.4	Another C++ Program: Adding Integers	43
2.5	Memory Concepts	47
2.6	Arithmetic	48
2.7	Decision Making: Equality and Relational Operators	51
2.8	Wrap-Up	55

3	Introduction to Classes, Objects and Strings	64
3.1	Introduction	65
3.2	Defining a Class with a Member Function	65
3.3	Defining a Member Function with a Parameter	68
3.4	Data Members, <i>set</i> Functions and <i>get</i> Functions	71
3.5	Initializing Objects with Constructors	77
3.6	Placing a Class in a Separate File for Reusability	81
3.7	Separating Interface from Implementation	84
3.8	Validating Data with <i>set</i> Functions	90
3.9	Wrap-Up	95
4	Control Statements: Part 1	101
4.1	Introduction	102
4.2	Algorithms	102
4.3	Pseudocode	103
4.4	Control Structures	104
4.5	<i>if</i> Selection Statement	107
4.6	<i>if...else</i> Double-Selection Statement	108
4.7	<i>while</i> Repetition Statement	113
4.8	Formulating Algorithms: Counter-Controlled Repetition	114
4.9	Formulating Algorithms: Sentinel-Controlled Repetition	120
4.10	Formulating Algorithms: Nested Control Statements	130
4.11	Assignment Operators	134
4.12	Increment and Decrement Operators	135
4.13	Wrap-Up	138
5	Control Statements: Part 2	152
5.1	Introduction	153
5.2	Essentials of Counter-Controlled Repetition	153
5.3	<i>for</i> Repetition Statement	155
5.4	Examples Using the <i>for</i> Statement	158
5.5	<i>do...while</i> Repetition Statement	162
5.6	<i>switch</i> Multiple-Selection Statement	164
5.7	<i>break</i> and <i>continue</i> Statements	173
5.8	Logical Operators	174
5.9	Confusing the Equality (<i>==</i>) and Assignment (<i>=</i>) Operators	179
5.10	Structured Programming Summary	180
5.11	Wrap-Up	185

6	Functions and an Introduction to Recursion	194
6.1	Introduction	195
6.2	Program Components in C++	196
6.3	Math Library Functions	197
6.4	Function Definitions with Multiple Parameters	198
6.5	Function Prototypes and Argument Coercion	203
6.6	C++ Standard Library Headers	205
6.7	Case Study: Random Number Generation	207
6.8	Case Study: Game of Chance; Introducing enum	212
6.9	Storage Classes	215
6.10	Scope Rules	218
6.11	Function Call Stack and Activation Records	221
6.12	Functions with Empty Parameter Lists	225
6.13	Inline Functions	225
6.14	References and Reference Parameters	227
6.15	Default Arguments	231
6.16	Unary Scope Resolution Operator	232
6.17	Function Overloading	234
6.18	Function Templates	236
6.19	Recursion	239
6.20	Example Using Recursion: Fibonacci Series	242
6.21	Recursion vs. Iteration	245
6.22	Wrap-Up	248
7	Arrays and Vectors	267
7.1	Introduction	268
7.2	Arrays	269
7.3	Declaring Arrays	270
7.4	Examples Using Arrays	271
7.4.1	Declaring an Array and Using a Loop to Initialize the Array's Elements	271
7.4.2	Initializing an Array in a Declaration with an Initializer List	272
7.4.3	Specifying an Array's Size with a Constant Variable and Setting Array Elements with Calculations	273
7.4.4	Summing the Elements of an Array	275
7.4.5	Using Bar Charts to Display Array Data Graphically	276
7.4.6	Using the Elements of an Array as Counters	277
7.4.7	Using Arrays to Summarize Survey Results	278
7.4.8	Static Local Arrays and Automatic Local Arrays	281

x **Contents**

7.5	Passing Arrays to Functions	283
7.6	Case Study: Class <code>GradeBook</code> Using an Array to Store Grades	287
7.7	Searching Arrays with Linear Search	293
7.8	Sorting Arrays with Insertion Sort	294
7.9	Multidimensional Arrays	297
7.10	Case Study: Class <code>GradeBook</code> Using a Two-Dimensional Array	300
7.11	Introduction to C++ Standard Library Class Template <code>vector</code>	307
7.12	Wrap-Up	313

8 **Pointers** **330**

8.1	Introduction	331
8.2	Pointer Variable Declarations and Initialization	331
8.3	Pointer Operators	332
8.4	Pass-by-Reference with Pointers	335
8.5	Using <code>const</code> with Pointers	339
8.6	Selection Sort Using Pass-by-Reference	343
8.7	<code>sizeof</code> Operator	347
8.8	Pointer Expressions and Pointer Arithmetic	349
8.9	Relationship Between Pointers and Arrays	352
8.10	Pointer-Based String Processing	354
8.11	Arrays of Pointers	357
8.12	Function Pointers	358
8.13	Wrap-Up	361

9 **Classes: A Deeper Look, Part 1** **379**

9.1	Introduction	380
9.2	Time Class Case Study	381
9.3	Class Scope and Accessing Class Members	388
9.4	Separating Interface from Implementation	389
9.5	Access Functions and Utility Functions	390
9.6	Time Class Case Study: Constructors with Default Arguments	393
9.7	Destructors	398
9.8	When Constructors and Destructors Are Called	399
9.9	Time Class Case Study: A Subtle Trap—Returning a Reference to a <code>private</code> Data Member	402
9.10	Default Memberwise Assignment	405
9.11	Wrap-Up	407

10 **Classes: A Deeper Look, Part 2** **414**

10.1	Introduction	415
------	--------------	-----

10.2	<code>const</code> (Constant) Objects and <code>const</code> Member Functions	415
10.3	Composition: Objects as Members of Classes	423
10.4	<code>friend</code> Functions and <code>friend</code> Classes	429
10.5	Using the <code>this</code> Pointer	431
10.6	<code>static</code> Class Members	436
10.7	Proxy Classes	441
10.8	Wrap-Up	445

11 Operator Overloading; Class `string` **451**

11.1	Introduction	452
11.2	Using the Overloaded Operators of Standard Library Class <code>string</code>	453
11.3	Fundamentals of Operator Overloading	456
11.4	Overloading Binary Operators	457
11.5	Overloading the Binary Stream Insertion and Stream Extraction Operators	458
11.6	Overloading Unary Operators	462
11.7	Overloading the Unary Prefix and Postfix <code>++</code> and <code>--</code> Operators	463
11.8	Case Study: A <code>Date</code> Class	464
11.9	Dynamic Memory Management	469
11.10	Case Study: <code>Array</code> Class	471
	11.10.1 Using the <code>Array</code> Class	472
	11.10.2 <code>Array</code> Class Definition	475
11.11	Operators as Member Functions vs. Non-Member Functions	483
11.12	Converting between Types	483
11.13	<code>explicit</code> Constructors	485
11.14	Building a <code>String</code> Class	487
11.15	Wrap-Up	488

12 Object-Oriented Programming: Inheritance **499**

12.1	Introduction	500
12.2	Base Classes and Derived Classes	500
12.3	<code>protected</code> Members	503
12.4	Relationship between Base Classes and Derived Classes	503
	12.4.1 Creating and Using a <code>CommissionEmployee</code> Class	504
	12.4.2 Creating a <code>BasePlusCommissionEmployee</code> Class Without Using Inheritance	508
	12.4.3 Creating a <code>CommissionEmployee</code> – <code>BasePlusCommissionEmployee</code> Inheritance Hierarchy	514
	12.4.4 <code>CommissionEmployee</code> – <code>BasePlusCommissionEmployee</code> Inheritance Hierarchy Using <code>protected</code> Data	519

12.4.5	CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using <code>private</code> Data	522
12.5	Constructors and Destructors in Derived Classes	527
12.6	<code>public</code> , <code>protected</code> and <code>private</code> Inheritance	527
12.7	Software Engineering with Inheritance	528
12.8	Wrap-Up	529

13 Object-Oriented Programming: Polymorphism 534

13.1	Introduction	535
13.2	Introduction to Polymorphism: Polymorphic Video Game	536
13.3	Relationships Among Objects in an Inheritance Hierarchy	536
13.3.1	Invoking Base-Class Functions from Derived-Class Objects	537
13.3.2	Aiming Derived-Class Pointers at Base-Class Objects	540
13.3.3	Derived-Class Member-Function Calls via Base-Class Pointers	541
13.3.4	Virtual Functions	543
13.4	Type Fields and <code>switch</code> Statements	549
13.5	Abstract Classes and Pure <code>virtual</code> Functions	549
13.6	Case Study: Payroll System Using Polymorphism	551
13.6.1	Creating Abstract Base Class <code>Employee</code>	552
13.6.2	Creating Concrete Derived Class <code>SalariedEmployee</code>	556
13.6.3	Creating Concrete Derived Class <code>CommissionEmployee</code>	558
13.6.4	Creating Indirect Concrete Derived Class <code>BasePlusCommissionEmployee</code>	560
13.6.5	Demonstrating Polymorphic Processing	562
13.7	(Optional) Polymorphism, Virtual Functions and Dynamic Binding “Under the Hood”	566
13.8	Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, <code>dynamic_cast</code> , <code>typeid</code> and <code>type_info</code>	569
13.9	Virtual Destructors	573
13.10	Wrap-Up	573

14 Templates 579

14.1	Introduction	580
14.2	Function Templates	580
14.3	Overloading Function Templates	583
14.4	Class Templates	584

14.5	Nontype Parameters and Default Types for Class Templates	590
14.6	Wrap-Up	591

15 Stream Input/Output **595**

15.1	Introduction	596
15.2	Streams	597
15.2.1	Classic Streams vs. Standard Streams	597
15.2.2	<code>iostream</code> Library Headers	598
15.2.3	Stream Input/Output Classes and Objects	598
15.3	Stream Output	601
15.3.1	Output of <code>char *</code> Variables	601
15.3.2	Character Output Using Member Function <code>put</code>	601
15.4	Stream Input	602
15.4.1	<code>get</code> and <code>getline</code> Member Functions	602
15.4.2	<code>istream</code> Member Functions <code>peek</code> , <code>putback</code> and <code>ignore</code>	605
15.4.3	Type-Safe I/O	605
15.5	Unformatted I/O Using <code>read</code> , <code>write</code> and <code>gcount</code>	605
15.6	Introduction to Stream Manipulators	606
15.6.1	Integral Stream Base: <code>dec</code> , <code>oct</code> , <code>hex</code> and <code>setbase</code>	607
15.6.2	Floating-Point Precision (<code>precision</code> , <code>setprecision</code>)	607
15.6.3	Field Width (<code>width</code> , <code>setw</code>)	609
15.6.4	User-Defined Output Stream Manipulators	610
15.7	Stream Format States and Stream Manipulators	612
15.7.1	Trailing Zeros and Decimal Points (<code>showpoint</code>)	612
15.7.2	Justification (<code>left</code> , <code>right</code> and <code>internal</code>)	613
15.7.3	Padding (<code>fill</code> , <code>setfill</code>)	615
15.7.4	Integral Stream Base (<code>dec</code> , <code>oct</code> , <code>hex</code> , <code>showbase</code>)	616
15.7.5	Floating-Point Numbers; Scientific and Fixed Notation (<code>scientific</code> , <code>fixed</code>)	617
15.7.6	Uppercase/Lowercase Control (<code>uppercase</code>)	618
15.7.7	Specifying Boolean Format (<code>boolalpha</code>)	618
15.7.8	Setting and Resetting the Format State via Member Function flags	619
15.8	Stream Error States	620
15.9	Tying an Output Stream to an Input Stream	622
15.10	Wrap-Up	623

16 Exception Handling: A Deeper Look **632**

16.1	Introduction	633
------	--------------	-----

16.2	Example: Handling an Attempt to Divide by Zero	633
16.3	When to Use Exception Handling	639
16.4	Rethrowing an Exception	640
16.5	Exception Specifications	641
16.6	Processing Unexpected Exceptions	642
16.7	Stack Unwinding	642
16.8	Constructors, Destructors and Exception Handling	644
16.9	Exceptions and Inheritance	645
16.10	Processing new Failures	645
16.11	Class <code>unique_ptr</code> and Dynamic Memory Allocation	648
16.12	Standard Library Exception Hierarchy	650
16.13	Wrap-Up	652

17 File Processing **658**

17.1	Introduction	659
17.2	Files and Streams	659
17.3	Creating a Sequential File	660
17.4	Reading Data from a Sequential File	664
17.5	Updating Sequential Files	669
17.6	Random-Access Files	670
17.7	Creating a Random-Access File	671
17.8	Writing Data Randomly to a Random-Access File	675
17.9	Reading from a Random-Access File Sequentially	677
17.10	Case Study: A Transaction-Processing Program	679
17.11	Object Serialization	686
17.12	Wrap-Up	686

18 Class `string` and String Stream Processing **696**

18.1	Introduction	697
18.2	<code>string</code> Assignment and Concatenation	698
18.3	Comparing <code>strings</code>	700
18.4	Substrings	703
18.5	Swapping <code>strings</code>	703
18.6	<code>string</code> Characteristics	704
18.7	Finding Substrings and Characters in a <code>string</code>	706
18.8	Replacing Characters in a <code>string</code>	708
18.9	Inserting Characters into a <code>string</code>	710
18.10	Conversion to C-Style Pointer-Based <code>char *</code> Strings	711
18.11	Iterators	713

18.12	String Stream Processing	714
18.13	Wrap-Up	717
19	Searching and Sorting	724
19.1	Introduction	725
19.2	Searching Algorithms	725
19.2.1	Efficiency of Linear Search	726
19.2.2	Binary Search	727
19.3	Sorting Algorithms	732
19.3.1	Efficiency of Selection Sort	733
19.3.2	Efficiency of Insertion Sort	733
19.3.3	Merge Sort (A Recursive Implementation)	733
19.4	Wrap-Up	740
20	Custom Templated Data Structures	746
20.1	Introduction	747
20.2	Self-Referential Classes	748
20.3	Dynamic Memory Allocation and Data Structures	749
20.4	Linked Lists	749
20.5	Stacks	764
20.6	Queues	768
20.7	Trees	772
20.8	Wrap-Up	780
21	Bits, Characters, C Strings and structs	791
21.1	Introduction	792
21.2	Structure Definitions	792
21.3	typedef	794
21.4	Example: Card Shuffling and Dealing Simulation	794
21.5	Bitwise Operators	797
21.6	Bit Fields	806
21.7	Character-Handling Library	810
21.8	Pointer-Based String Manipulation Functions	815
21.9	Pointer-Based String-Conversion Functions	822
21.10	Search Functions of the Pointer-Based String-Handling Library	827
21.11	Memory Functions of the Pointer-Based String-Handling Library	831
21.12	Wrap-Up	835

22	Standard Template Library (STL)	850
22.1	Introduction to the Standard Template Library (STL)	851
22.2	Introduction to Containers	853
22.3	Introduction to Iterators	856
22.4	Introduction to Algorithms	861
22.5	Sequence Containers	863
22.5.1	vector Sequence Container	864
22.5.2	list Sequence Container	871
22.5.3	deque Sequence Container	875
22.6	Associative Containers	877
22.6.1	multiset Associative Container	877
22.6.2	set Associative Container	880
22.6.3	multimap Associative Container	881
22.6.4	map Associative Container	883
22.7	Container Adapters	885
22.7.1	stack Adapter	885
22.7.2	queue Adapter	887
22.7.3	priority_queue Adapter	888
22.8	Algorithms	890
22.8.1	fill, fill_n, generate and generate_n	890
22.8.2	equal, mismatch and lexicographical_compare	892
22.8.3	remove, remove_if, remove_copy and remove_copy_if	895
22.8.4	replace, replace_if, replace_copy and replace_copy_if	897
22.8.5	Mathematical Algorithms	900
22.8.6	Basic Searching and Sorting Algorithms	903
22.8.7	swap, iter_swap and swap_ranges	905
22.8.8	copy_backward, merge, unique and reverse	906
22.8.9	inplace_merge, unique_copy and reverse_copy	909
22.8.10	Set Operations	910
22.8.11	lower_bound, upper_bound and equal_range	913
22.8.12	Heapsort	915
22.8.13	min and max	918
22.8.14	STL Algorithms Not Covered in This Chapter	919
22.9	Class bitset	920
22.10	Function Objects	924
22.11	Wrap-Up	927
23	Boost Libraries, Technical Report I and C++0x	936
23.1	Introduction	937

23.2	Deitel Online C++ and Related Resource Centers	937
23.3	Boost Libraries	937
23.4	Boost Libraries Overview	938
23.5	Regular Expressions with the <code>regex</code> Library	941
23.5.1	Regular Expression Example	942
23.5.2	Validating User Input with Regular Expressions	944
23.5.3	Replacing and Splitting Strings	947
23.6	Smart Pointers	950
23.6.1	Reference Counted <code>shared_ptr</code>	950
23.6.2	<code>weak_ptr</code> : <code>shared_ptr</code> Observer	954
23.7	Technical Report 1	960
23.8	C++0x	961
23.9	Core Language Changes	962
23.10	Wrap-Up	967
24	Other Topics	974
24.1	Introduction	975
24.2	<code>const_cast</code> Operator	975
24.3	<code>mutable</code> Class Members	977
24.4	namespaces	979
24.5	Operator Keywords	982
24.6	Pointers to Class Members (<code>.</code> , <code>*</code> and <code>->*</code>)	984
24.7	Multiple Inheritance	986
24.8	Multiple Inheritance and <code>virtual</code> Base Classes	991
24.9	Wrap-Up	996
	Chapters on the Web	1001
A	Operator Precedence and Associativity	1002
B	ASCII Character Set	1004
C	Fundamental Types	1005
D	Number Systems	1007
D.1	Introduction	1008

D.2	Abbreviating Binary Numbers as Octal and Hexadecimal Numbers	1011
D.3	Converting Octal and Hexadecimal Numbers to Binary Numbers	1012
D.4	Converting from Binary, Octal or Hexadecimal to Decimal	1012
D.5	Converting from Decimal to Binary, Octal or Hexadecimal	1013
D.6	Negative Binary Numbers: Two's Complement Notation	1015

E Preprocessor 1020

E.1	Introduction	1021
E.2	#include Preprocessor Directive	1021
E.3	#define Preprocessor Directive: Symbolic Constants	1022
E.4	#define Preprocessor Directive: Macros	1022
E.5	Conditional Compilation	1024
E.6	#error and #pragma Preprocessor Directives	1025
E.7	Operators # and ##	1026
E.8	Predefined Symbolic Constants	1026
E.9	Assertions	1027
E.10	Wrap-Up	1027

Appendices on the Web 1033

Index 1035

Chapters 25–26 and Appendices F–I are PDF documents posted online at the book's Companion Website, which is accessible from www.pearsonhighered.com/deitel.

25 ATM Case Study, Part I: Object-Oriented Design with the UML 25-1

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

26	ATM Case Study, Part 2: Implementing an Object-Oriented Design	26-1
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 exit and atexit	F-9
F.7	Type Qualifier volatile	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 calloc and realloc	F-13
F.11	Unconditional Branch: goto	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

H	Using the Visual Studio Debugger	H-1
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
I	Using the GNU C++ Debugger	I-1
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