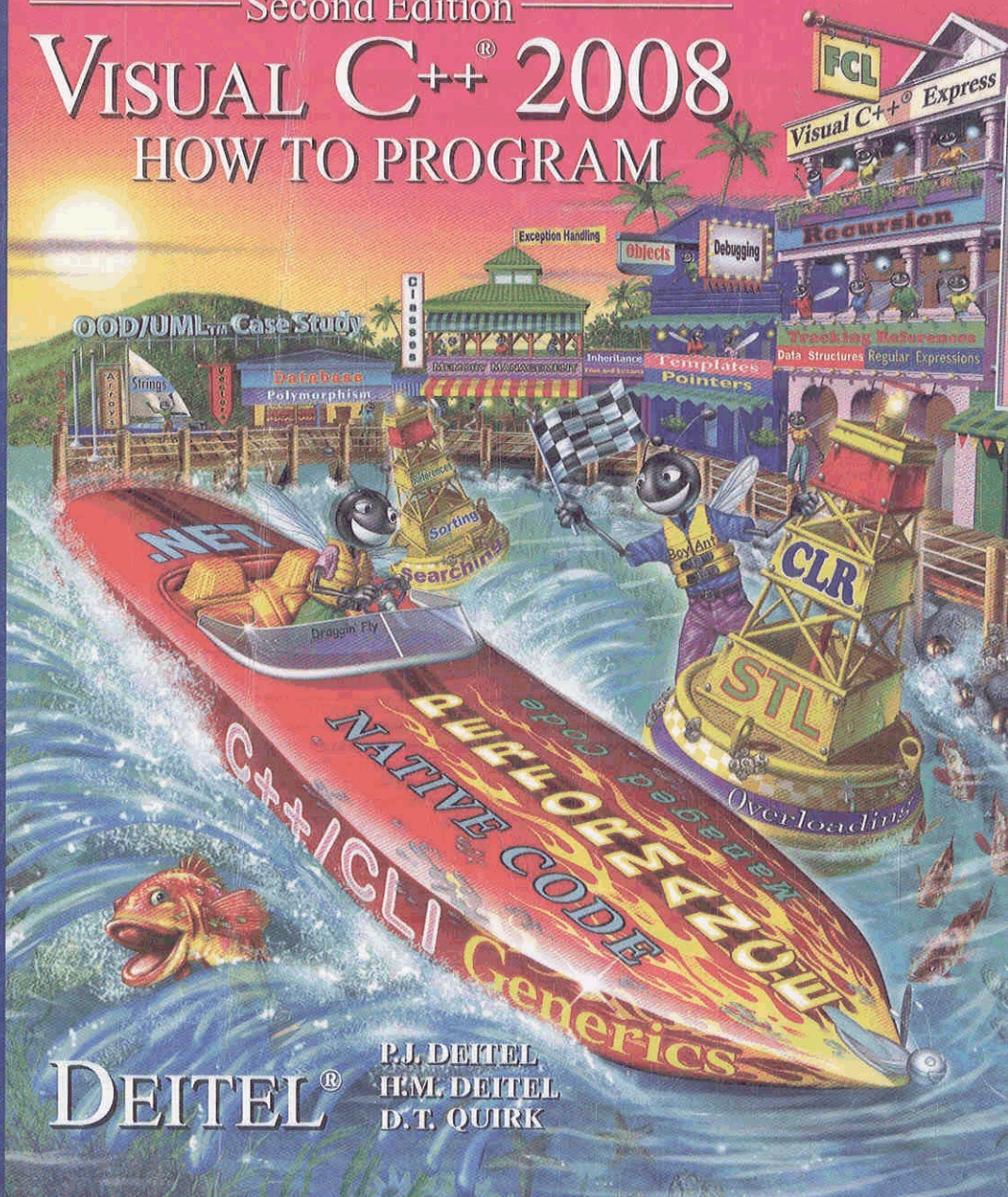


Pearson International Edition

Second Edition

VISUAL C++[®] 2008

HOW TO PROGRAM



DEITEL[®]

P.J. DEITEL
H.M. DEITEL
D.T. QUIRK

Contents

Preface **xxii**

Before You Begin **xxxvii**

I Introduction to Computers, the Internet and Visual C++ **I**

1.1	Introduction	2
1.2	What Is a Computer?	3
1.3	Computer Organization	4
1.4	Early Operating Systems	5
1.5	Personal Computing, Distributed Computing and Client/Server Computing	5
1.6	The Internet and the World Wide Web	6
1.7	Hardware Trends	6
1.8	Microsoft's Windows® Operating System	7
1.9	Machine Languages, Assembly Languages and High-Level Languages	7
1.10	Visual C++	9
1.11	C++ Standard Library	12
1.12	Java, C# and Visual Basic	13
1.13	Other High-Level Languages	14
1.14	Microsoft's .NET	15
1.15	The .NET Framework and the Common Language Runtime	16
1.16	Key Software Trend: Object Technology	17
1.17	Typical Visual C++ Development Life Cycle	19
1.18	Test-Driving a Visual C++ Application	21
1.19	Software Technologies	24
1.20	Future of Visual C++: Open Source Boost Libraries, TR1 and C++0x	25
1.21	(Only Required Section of the Case Study) Software Engineering Case Study: Introduction to Object Technology and the UML	26
1.22	Wrap-Up	31

2 Dive Into® Visual C++® 2008 Express **41**

2.1	Introduction	42
2.2	Overview of the Visual Studio 2008 IDE	42
2.3	Menu Bar and Toolbars	49
2.4	Navigating the Visual Studio 2008 IDE	52
2.4.1	Solution Explorer	54
2.4.2	Properties Window	55

2.5	Using Help	56
2.6	Wrap-Up	56
3	Introduction to Visual C++ Programming	61
3.1	Introduction	62
3.2	First Program in Visual C++: Printing a Line of Text	62
3.3	Modifying Our First Visual C++ Program	66
3.4	Another Visual C++ Program: Adding Integers	67
3.5	Memory Concepts	72
3.6	Arithmetic	73
3.7	Decision Making: Equality and Relational Operators	76
3.8	(Optional) Software Engineering Case Study: Examining the ATM Requirements Specification	81
3.9	Wrap-Up	91
4	Introduction to Classes and Objects	101
4.1	Introduction	102
4.2	Classes, Objects, Member Functions and Data Members	102
4.3	Overview of the Chapter Examples	104
4.4	Defining a Class with a Member Function	105
4.5	Defining a Member Function with a Parameter	108
4.6	Data Members, <i>set</i> Functions and <i>get</i> Functions	111
4.7	Initializing Objects with Constructors	118
4.8	Placing a Class in a Separate File for Reusability	122
4.9	Separating Interface from Implementation	126
4.10	Validating Data with <i>set</i> Functions	132
4.11	Defining a Managed Class with Member Functions in C++/CLI	137
4.12	Instance Variables and Properties in C++/CLI	140
4.13	(Optional) Software Engineering Case Study: Identifying the Classes in the ATM Requirements Specification	143
4.14	Wrap-Up	151
5	Control Statements: Part I	159
5.1	Introduction	160
5.2	Algorithms	160
5.3	Pseudocode	161
5.4	Control Structures	162
5.5	<i>if</i> Selection Statement	166
5.6	<i>if...else</i> Double-Selection Statement	168
5.7	<i>while</i> Repetition Statement	173
5.8	Formulating Algorithms: Counter-Controlled Repetition	174
5.9	Formulating Algorithms: Sentinel-Controlled Repetition	181
5.10	Formulating Algorithms: Nested Control Statements	191
5.11	Assignment Operators	196
5.12	Increment and Decrement Operators	197

5.13	(Optional) Software Engineering Case Study: Identifying Class Attributes in the ATM System	200
5.14	Wrap-Up	205

6 Control Statements: Part 2 220

6.1	Introduction	221
6.2	Essentials of Counter-Controlled Repetition	221
6.3	for Repetition Statement	223
6.4	Examples Using the for Statement	228
6.5	do...while Repetition Statement	232
6.6	switch Multiple-Selection Statement	234
6.7	break and continue Statements	244
6.8	Logical Operators	246
6.9	Confusing the Equality (==) and Assignment (=) Operators	251
6.10	Structured Programming Summary	252
6.11	(Optional) Software Engineering Case Study: Identifying Objects' States and Activities in the ATM System	257
6.12	Wrap-Up	261

7 Functions and an Introduction to Recursion 273

7.1	Introduction	274
7.2	Program Components in Visual C++	275
7.3	Math Library Functions	277
7.4	Function Definitions with Multiple Parameters	278
7.5	Function Prototypes and Argument Coercion	283
7.6	C++ Standard Library Header Files	285
7.7	Case Study: Random Number Generation	287
7.8	Case Study: Game of Chance; Introducing enum	293
7.9	Storage Classes	297
7.10	Scope Rules	299
7.11	Function-Call Stack and Activation Records	303
7.12	Functions with Empty Parameter Lists	307
7.13	Inline Functions	308
7.14	References and Reference Parameters	309
7.15	Default Arguments	313
7.16	Unary Scope Resolution Operator	315
7.17	Function Overloading	317
7.18	Function Templates	319
7.19	Recursion	321
7.20	Example Using Recursion: Fibonacci Series	325
7.21	Recursion vs. Iteration	328
7.22	Enumerations in C++/CLI	331
7.23	(Optional) Software Engineering Case Study: Identifying Class Operations in the ATM System	332
7.24	Wrap-Up	339

8	Arrays and Vectors	361
8.1	Introduction	362
8.2	Arrays	363
8.3	Declaring Arrays	365
8.4	Examples Using Arrays	365
8.4.1	Declaring an Array and Using a Loop to Initialize the Array's Elements	366
8.4.2	Initializing an Array in a Declaration with an Initializer List	366
8.4.3	Specifying an Array's Size with a Constant Variable and Setting Array Elements with Calculations	368
8.4.4	Summing the Elements of an Array	371
8.4.5	Using Bar Charts to Display Array Data Graphically	371
8.4.6	Using the Elements of an Array as Counters	373
8.4.7	Using Arrays to Summarize Survey Results	374
8.4.8	Static Local Arrays and Automatic Local Arrays	377
8.5	Passing Arrays to Functions	379
8.6	Case Study: Class <code>GradeBook</code> Using an Array to Store Grades	384
8.7	Searching Arrays with Linear Search	390
8.8	Sorting Arrays with Insertion Sort	392
8.9	Multidimensional Arrays	394
8.10	Case Study: Class <code>GradeBook</code> Using a Two-Dimensional Array	397
8.11	Introduction to C++ Standard Library Class Template <code>vector</code>	404
8.12	Introduction to Managed Arrays with C++/CLI	409
8.13	<code>for each</code> Statement	415
8.14	Multidimensional Arrays in C++/CLI	416
8.15	(Optional) Software Engineering Case Study: Collaboration Among Objects in the ATM System	422
8.16	Wrap-Up	429
9	Pointers and Pointer-Based Strings	449
9.1	Introduction	450
9.2	Pointer Variable Declarations and Initialization	451
9.3	Pointer Operators	452
9.4	Passing Arguments to Functions by Reference with Pointers	455
9.5	Using <code>const</code> with Pointers	459
9.6	Selection Sort Using Pass-by-Reference	466
9.7	<code>sizeof</code> Operator	469
9.8	Pointer Expressions and Pointer Arithmetic	472
9.9	Relationship Between Pointers and Arrays	475
9.10	Arrays of Pointers	479
9.11	Case Study: Card Shuffling and Dealing Simulation	480
9.12	Function Pointers	486
9.13	Introduction to Pointer-Based String Processing	491
9.13.1	Fundamentals of Characters and Pointer-Based Strings	492
9.13.2	String-Manipulation Functions of the String-Handling Library	494

9.14	Introduction to C++/CLI Handles	502
9.15	Passing Arguments to Functions by Reference with Handles	507
9.16	Tracking References and References to Handles	508
9.17	Interior Pointers	512
9.18	Wrap-Up	512

10 Classes: A Deeper Look, Part 1 **541**

10.1	Introduction	542
10.2	Time Class Case Study	543
10.3	Class Scope and Accessing Class Members	549
10.4	Separating Interface from Implementation	552
10.5	Access Functions and Utility Functions	552
10.6	Time Class Case Study: Constructors with Default Arguments	555
10.7	Destructors	561
10.8	When Constructors and Destructors Are Called	562
10.9	Time Class Case Study: A Subtle Trap—Returning a Reference to a private Data Member	565
10.10	Default Memberwise Assignment	568
10.11	Class View and Object Browser	570
10.12	(Optional) Software Engineering Case Study: Starting to Program the Classes of the ATM System	572
10.13	Wrap-Up	579

11 Classes: A Deeper Look, Part 2 **586**

11.1	Introduction	587
11.2	const (Constant) Objects and const Member Functions	588
11.3	Composition: Objects as Members of Classes	597
11.4	friend Functions and friend Classes	604
11.5	Using the this Pointer	607
11.6	Dynamic Memory Management with Operators new and delete	613
11.7	static Class Members	615
11.8	Data Abstraction and Information Hiding	621
	11.8.1 Example: Array Abstract Data Type	623
	11.8.2 Example: String Abstract Data Type	623
	11.8.3 Example: Queue Abstract Data Type	624
11.9	Container Classes and Iterators	624
11.10	Proxy Classes	625
11.11	const and friend in C++/CLI	628
11.12	Dynamic Memory Management in C++/CLI	629
11.13	Stack Semantics in C++/CLI	630
11.14	Finalizers	632
11.15	Value Types vs. Reference Types in C++/CLI	632
11.16	Boxing and Unboxing in C++/CLI	634
11.17	Indexers	635
11.18	Wrap-Up	639

12 Operator Overloading; String and Array Objects **648**

12.1	Introduction	649
12.2	Fundamentals of Operator Overloading	650
12.3	Restrictions on Operator Overloading	651
12.4	Operator Functions as Class Members vs. Global Functions	653
12.5	Overloading Stream Insertion and Stream Extraction Operators	654
12.6	Overloading Unary Operators	658
12.7	Overloading Binary Operators	658
12.8	Case Study: Array Class	659
12.9	Converting between Types	671
12.10	Case Study: String Class	672
12.11	Overloading ++ and --	684
12.12	Case Study: A Date Class	686
12.13	Standard Library Class string	690
12.14	explicit Constructors	694
12.15	C++/CLI Operators and Constructors	698
12.16	Wrap-Up	698

13 Object-Oriented Programming: Inheritance **711**

13.1	Introduction	712
13.2	Base Classes and Derived Classes	713
13.3	protected Members	716
13.4	Relationship between Base Classes and Derived Classes	716
13.4.1	Creating and Using a CommissionEmployee Class	717
13.4.2	Creating a BasePlusCommissionEmployee Class Without Using Inheritance	722
13.4.3	Creating a CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy	728
13.4.4	CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using protected Data	733
13.4.5	CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using private Data	741
13.5	Constructors and Destructors in Derived Classes	748
13.6	public, protected and private Inheritance	757
13.7	Software Engineering with Inheritance	758
13.8	Inheritance in C++/CLI	759
13.9	Wrap-Up	759

14 Object-Oriented Programming: Polymorphism **765**

14.1	Introduction	766
14.2	Polymorphism Examples	768
14.3	Relationships Among Objects in an Inheritance Hierarchy	769
14.3.1	Invoking Base-Class Functions from Derived-Class Objects	769

14.3.2	Aiming Derived-Class Pointers at Base-Class Objects	777
14.3.3	Derived-Class Member-Function Calls via Base-Class Pointers	778
14.3.4	Virtual Functions	779
14.3.5	Summary of the Allowed Assignments Between Base-Class and Derived-Class Objects and Pointers	785
14.4	Type Fields and <code>switch</code> Statements	786
14.5	Abstract Classes and Pure virtual Functions	787
14.6	Case Study: Payroll System Using Polymorphism	789
14.6.1	Creating Abstract Base Class <code>Employee</code>	790
14.6.2	Creating Concrete Derived Class <code>SalariesEmployee</code>	794
14.6.3	Creating Concrete Derived Class <code>HourlyEmployee</code>	796
14.6.4	Creating Concrete Derived Class <code>CommissionEmployee</code>	798
14.6.5	Creating Indirect Concrete Derived Class <code>BasePlusCommissionEmployee</code>	800
14.6.6	Demonstrating Polymorphic Processing	802
14.7	(Optional) Polymorphism, Virtual Functions and Dynamic Binding “Under the Hood”	806
14.8	Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, <code>dynamic_cast</code> , <code>typeid</code> and <code>typeid</code>	810
14.9	Virtual Destructors	814
14.10	Polymorphism in C++/CLI	815
14.11	(Optional) Software Engineering Case Study: Incorporating Inheritance into the ATM System	818
14.12	Wrap-Up	826

15 Templates and Generics **833**

15.1	Introduction	834
15.2	Function Templates	835
15.3	Overloading Function Templates	838
15.4	Class Templates	839
15.5	Nontype Parameters and Default Types for Class Templates	845
15.6	Notes on Templates and Inheritance	846
15.7	Notes on Templates and Friends	846
15.8	Notes on Templates and <code>static</code> Members	848
15.9	Templates in C++/CLI	848
15.10	.NET Generics in C++/CLI	848
15.11	Generic Type Constraints	853
15.12	Contrasting Templates and Generics	855
15.13	Wrap-Up	857

16 Exception Handling **864**

16.1	Introduction	865
16.2	Exception-Handling Overview	866
16.3	Example: Divide by Zero Without Exception Handling	867
16.4	Example: Handling an Attempt to Divide by Zero	869

16.5	When to Use Exception Handling	875
16.6	Rethrowing an Exception	876
16.7	Processing Unexpected Exceptions	877
16.8	Stack Unwinding	878
16.9	Constructors, Destructors and Exception Handling	880
16.10	Exceptions and Inheritance	880
16.11	Processing new Failures	881
16.12	Class <code>auto_ptr</code> and Dynamic Memory Allocation	884
16.13	Standard Library Exception Hierarchy	887
16.14	Other Error-Handling Techniques	888
16.15	.NET Exception Hierarchy with C++/CLI	889
	16.15.1 Classes <code>ApplicationException</code> and <code>SystemException</code>	889
	16.15.2 Determining Which Exceptions a Function Throws	890
16.16	<code>finally</code> Block in C++/CLI	891
16.17	Exception Properties in C++/CLI	899
16.18	User-Defined Exception Classes in .NET	904
16.19	Wrap-Up	905

17 Stream Input/Output and Files **913**

17.1	Introduction	914
17.2	Streams	915
	17.2.1 Classic Streams vs. Standard Streams	916
	17.2.2 <code>iostream</code> Library Header Files	916
	17.2.3 Stream-Input/Output Classes and Objects	917
17.3	Stream Output	919
	17.3.1 Output of <code>char *</code> Variables	920
	17.3.2 Character Output Using Member Function <code>put</code>	920
17.4	Stream Input	921
	17.4.1 <code>get</code> and <code>getline</code> Member Functions	921
	17.4.2 <code>istream</code> Member Functions <code>peek</code> , <code>putback</code> and <code>ignore</code>	924
	17.4.3 Type-Safe I/O	924
17.5	Unformatted I/O Using <code>read</code> , <code>write</code> and <code>gcount</code>	925
17.6	Introduction to Stream Manipulators	926
	17.6.1 Integral Stream Base: <code>dec</code> , <code>oct</code> , <code>hex</code> and <code>setbase</code>	926
	17.6.2 Floating-Point Precision (<code>precision</code> , <code>setprecision</code>)	927
	17.6.3 Field Width (<code>width</code> , <code>setw</code>)	929
	17.6.4 User-Defined Output Stream Manipulators	930
17.7	Stream Format States and Stream Manipulators	931
	17.7.1 Trailing Zeros and Decimal Points (<code>showpoint</code>)	932
	17.7.2 Justification (<code>left</code> , <code>right</code> and <code>internal</code>)	933
	17.7.3 Padding (<code>fill</code> , <code>setfill</code>)	935
	17.7.4 Integral Stream Base (<code>dec</code> , <code>oct</code> , <code>hex</code> , <code>showbase</code>)	936
	17.7.5 Floating-Point Numbers; Scientific and Fixed Notation (<code>scientific</code> , <code>fixed</code>)	937
	17.7.6 Uppercase/Lowercase Control (<code>uppercase</code>)	938

17.7.7	Specifying Boolean Format (boolalpha)	939
17.7.8	Setting and Resetting the Format State via Member Function flags	940
17.8	Stream Error States	942
17.9	Tying an Output Stream to an Input Stream	944
17.10	Data Hierarchy	944
17.11	Files and Streams	946
17.12	Creating a Sequential File	947
17.13	Reading Data from a Sequential File	951
17.14	Updating Sequential Files	957
17.15	Wrap-Up	958

18 Files and Streams in .NET 975

18.1	Introduction	976
18.2	Files and Streams	976
18.3	Classes File and Directory	977
18.4	Creating a Sequential-Access Text File	981
18.5	Reading Data from a Sequential-Access Text File	986
18.6	Serialization	992
18.7	Creating a Sequential-Access File Using Object Serialization	992
18.8	Reading and Deserializing Data from a Sequential-Access Text File	996
18.9	Wrap-Up	997

19 Class string and String Stream Processing 1004

19.1	Introduction	1005
19.2	string Assignment and Concatenation	1007
19.3	Comparing strings	1009
19.4	Substrings	1012
19.5	Swapping strings	1012
19.6	string Characteristics	1013
19.7	Finding Substrings and Characters in a string	1016
19.8	Replacing Characters in a string	1018
19.9	Inserting Characters into a string	1020
19.10	Conversion to C-Style Pointer-Based char * Strings	1021
19.11	Iterators	1022
19.12	String Stream Processing	1024
19.13	Fundamentals of Characters and Strings in C++/CLI	1027
19.14	String Constructors	1027
19.15	String Indexer, Length Property and CopyTo Function	1029
19.16	Comparing Strings	1030
19.17	Locating Characters and Substrings in Strings	1033
19.18	Extracting Substrings from Strings	1036
19.19	Concatenating Strings	1037
19.20	Miscellaneous String Functions	1038
19.21	Class StringBuilder	1040
19.22	StringBuilder Class Length, Capacity, EnsureCapacity and Indexer	1041

19.23	StringBuiLder Class Append and AppendFormat Functions	1043
19.24	StringBuiLder Class Insert, Remove and Replace Functions	1045
19.25	Char Functions	1048
19.26	Wrap-Up	1050

20 Searching and Sorting **1060**

20.1	Introduction	1061
20.2	Searching Algorithms	1062
	20.2.1 Efficiency of Linear Search	1062
	20.2.2 Binary Search	1064
20.3	Sorting Algorithms	1069
	20.3.1 Efficiency of Selection Sort	1069
	20.3.2 Efficiency of Insertion Sort	1070
	20.3.3 Merge Sort (A Recursive Implementation)	1070
20.4	Wrap-Up	1077

21 Data Structures **1083**

21.1	Introduction	1084
21.2	Self-Referential Classes	1085
21.3	Dynamic Memory Allocation and Data Structures	1086
21.4	Linked Lists	1086
21.5	Stacks	1101
21.6	Queues	1106
21.7	Trees	1110
21.8	Wrap-Up	1118

22 Bits, Characters, C Strings and structs **1142**

22.1	Introduction	1143
22.2	Structure Definitions	1143
22.3	Initializing Structures	1146
22.4	Using Structures with Functions	1146
22.5	typedef	1146
22.6	Example: High-Performance Card Shuffling and Dealing Simulation	1147
22.7	Bitwise Operators	1150
22.8	Bit Fields	1159
22.9	Character-Handling Library Functions	1163
22.10	Pointer-Based String-Conversion Functions	1169
22.11	Search Functions of the Pointer-Based String-Handling Library	1174
22.12	Memory Functions of the Pointer-Based String-Handling Library	1179
22.13	Wrap-Up	1183

23 Standard Template Library (STL) **1195**

23.1	Introduction to the Standard Template Library (STL)	1197
------	---	------

23.1.1	Introduction to Containers	1198
23.1.2	Introduction to Iterators	1203
23.1.3	Introduction to Algorithms	1208
23.2	Sequence Containers	1210
23.2.1	vector Sequence Container	1210
23.2.2	list Sequence Container	1218
23.2.3	deque Sequence Container	1222
23.3	Associative Containers	1224
23.3.1	multiset Associative Container	1224
23.3.2	set Associative Container	1227
23.3.3	multimap Associative Container	1229
23.3.4	map Associative Container	1230
23.4	Container Adapters	1232
23.4.1	stack Adapter	1232
23.4.2	queue Adapter	1235
23.4.3	priority_queue Adapter	1236
23.5	Algorithms	1238
23.5.1	fill, fill_n, generate and generate_n	1238
23.5.2	equal, mismatch and lexicographical_compare	1240
23.5.3	remove, remove_if, remove_copy and remove_copy_if	1242
23.5.4	replace, replace_if, replace_copy and replace_copy_if	1245
23.5.5	Mathematical Algorithms	1247
23.5.6	Basic Searching and Sorting Algorithms	1251
23.5.7	swap, iter_swap and swap_ranges	1253
23.5.8	copy_backward, merge, unique and reverse	1255
23.5.9	inplace_merge, unique_copy and reverse_copy	1257
23.5.10	Set Operations	1259
23.5.11	lower_bound, upper_bound and equal_range	1262
23.5.12	Heapsort	1264
23.5.13	min and max	1267
23.5.14	STL Algorithms Not Covered in This Chapter	1267
23.6	Class bitset	1269
23.7	Function Objects	1273
23.8	Introduction to STL/CLR	1277
23.9	Wrap-Up	1278
23.10	STL Web Resources	1278

24 Regular Expressions

1289

24.1	Introduction	1290
24.2	Simple Regular Expressions and Class Regex	1290
24.3	Complex Regular Expressions	1296
24.4	Validating User Input with Regular Expressions	1297
24.5	Regex Member Functions Replace and Split	1300
24.6	Wrap-Up	1302

25	Collections	1306
25.1	Introduction	1307
25.2	Collections Overview	1307
25.3	Class Array and Enumerators	1310
25.4	Nongeneric Collections	1313
25.4.1	Class ArrayList	1313
25.4.2	Class Stack	1318
25.4.3	Class Hashtable	1321
25.5	Generic Collections	1326
25.5.1	Generic Class SortedDictionary	1326
25.5.2	Generic Class LinkedList	1329
25.6	Wrap-Up	1333

26	Other Topics	1339
26.1	Introduction	1340
26.2	Other Cast Operators	1340
26.3	namespaces	1343
26.4	Operator Keywords	1347
26.5	mutable Class Members	1349
26.6	Pointers to Class Members (. * and ->*)	1351
26.7	Multiple Inheritance	1353
26.8	Multiple Inheritance and virtual Base Classes	1357
26.9	Variable-Length Argument Lists	1362
26.10	Using Command-Line Arguments	1366
26.11	Delegates and Events in .NET	1370
26.12	Wrap-Up	1374

A	Operator Precedence and Associativity Chart	1381
A.1	Operator Precedence	1381

B	ASCII Character Set	1384
----------	----------------------------	-------------

C	Fundamental Types	1385
----------	--------------------------	-------------

D	Number Systems	1387
D.1	Introduction	1388
D.2	Abbreviating Binary Numbers as Octal and Hexadecimal Numbers	1391
D.3	Converting Octal and Hexadecimal Numbers to Binary Numbers	1392
D.4	Converting from Binary, Octal or Hexadecimal to Decimal	1392
D.5	Converting from Decimal to Binary, Octal or Hexadecimal	1393
D.6	Negative Binary Numbers: Two's Complement Notation	1395

E	Preprocessor	1400
E.1	Introduction	1401
E.2	The <code>#include</code> Preprocessor Directive	1401
E.3	The <code>#define</code> Preprocessor Directive: Symbolic Constants	1402
E.4	The <code>#define</code> Preprocessor Directive: Macros	1402
E.5	Conditional Compilation	1404
E.6	The <code>#error</code> and <code>#pragma</code> Preprocessor Directives	1405
E.7	Operators <code>#</code> and <code>##</code>	1406
E.8	Predefined Symbolic Constants	1406
E.9	Assertions	1407
E.10	Wrap-Up	1407

F	ATM Case Study Code	1412
F.1	ATM Case Study Implementation	1412
F.2	Class <code>ATM</code>	1413
F.3	Class <code>Screen</code>	1420
F.4	Class <code>Keypad</code>	1421
F.5	Class <code>CashDispenser</code>	1422
F.6	Class <code>DepositSlot</code>	1424
F.7	Class <code>Account</code>	1425
F.8	Class <code>BankDatabase</code>	1427
F.9	Class <code>Transaction</code>	1431
F.10	Class <code>BalanceInquiry</code>	1433
F.11	Class <code>Withdrawal</code>	1435
F.12	Class <code>Deposit</code>	1440
F.13	Test Program <code>ATMCaseStudy.cpp</code>	1443
F.14	Wrap-Up	1443

G	UML 2: Additional Diagram Types	1444
G.1	Introduction	1444
G.2	Additional Diagram Types	1444

H	Using the Visual Studio Debugger	1446
H.1	Introduction	1447
H.2	Breakpoints and the <code>Continue</code> Command	1447
H.3	<code>Locals</code> and <code>Watch</code> Windows	1452
H.4	Controlling Execution Using the <code>Step Into</code> , <code>Step Over</code> , <code>Step Out</code> and <code>Continue</code> Commands	1455
H.5	<code>Autos</code> Window	1458
H.6	Wrap-Up	1459

Index	1462
--------------	-------------