



HOW TO PROGRAM

SEVENTH EDITION

Pearson International Edition

P. J. Deitel

Deitel & Associates, Inc.

H. M. Deitel

Deitel & Associates, Inc.



Upper Saddle River, New Jersey 07458

Chapters 23–27 and Appendices F–I are PDF documents posted on the
Companion Website (located at www.pearsonhighered.com/deitel).

Preface **23**

I Introduction to Computers, the Internet and the World Wide Web **37**

1.1	Introduction	38
1.2	Computers: Hardware and Software	39
1.3	Computer Organization	40
1.4	Personal, Distributed and Client/Server Computing	41
1.5	The Internet and the World Wide Web	42
1.6	Web 2.0	42
1.7	Machine Languages, Assembly Languages and High-Level Languages	43
1.8	History of C and C++	44
1.9	C++ Standard Library	45
1.10	History of Java	46
1.11	Fortran, COBOL, Pascal and Ada	47
1.12	BASIC, Visual Basic, Visual C++, C# and .NET	47
1.13	Key Software Trend: Object Technology	48
1.14	Typical C++ Development Environment	49
1.15	Notes About C++ and <i>C++ How to Program, 7/e</i>	51
1.16	Test-Driving a C++ Application	52
1.17	Software Technologies	58
1.18	Future of C++: Open Source Boost Libraries, TR1 and C++0x	59
1.19	Software Engineering Case Study: Introduction to Object Technology and the UML	60
1.20	Wrap-Up	64
1.21	Web Resources	65

2 Introduction to C++ Programming **75**

2.1	Introduction	76
2.2	First Program in C++: Printing a Line of Text	76
2.3	Modifying Our First C++ Program	80
2.4	Another C++ Program: Adding Integers	81

2.5	Memory Concepts	85
2.6	Arithmetic	86
2.7	Decision Making: Equality and Relational Operators	90
2.8	Wrap-Up	94

3 Introduction to Classes and Objects 104

3.1	Introduction	105
3.2	Classes, Objects, Member Functions and Data Members	105
3.3	Defining a Class with a Member Function	107
3.4	Defining a Member Function with a Parameter	110
3.5	Data Members, <i>set</i> Functions and <i>get</i> Functions	113
3.6	Initializing Objects with Constructors	120
3.7	Placing a Class in a Separate File for Reusability	123
3.8	Separating Interface from Implementation	127
3.9	Validating Data with <i>set</i> Functions	133
3.10	Wrap-Up	138

4 Control Statements: Part I 145

4.1	Introduction	146
4.2	Algorithms	146
4.3	Pseudocode	147
4.4	Control Structures	148
4.5	<i>if</i> Selection Statement	151
4.6	<i>if...else</i> Double-Selection Statement	153
4.7	<i>while</i> Repetition Statement	158
4.8	Formulating Algorithms: Counter-Controlled Repetition	159
4.9	Formulating Algorithms: Sentinel-Controlled Repetition	165
4.10	Formulating Algorithms: Nested Control Statements	175
4.11	Assignment Operators	180
4.12	Increment and Decrement Operators	180
4.13	Wrap-Up	184

5 Control Statements: Part 2 199

5.1	Introduction	200
5.2	Essentials of Counter-Controlled Repetition	200
5.3	<i>for</i> Repetition Statement	202
5.4	Examples Using the <i>for</i> Statement	206
5.5	<i>do...while</i> Repetition Statement	210
5.6	<i>switch</i> Multiple-Selection Statement	212
5.7	<i>break</i> and <i>continue</i> Statements	221
5.8	Logical Operators	223
5.9	Confusing the Equality (<i>==</i>) and Assignment (<i>=</i>) Operators	227
5.10	Structured Programming Summary	228
5.11	Wrap-Up	233

6 Functions and an Introduction to Recursion 243

6.1	Introduction	244
6.2	Program Components in C++	245
6.3	Math Library Functions	246
6.4	Function Definitions with Multiple Parameters	247
6.5	Function Prototypes and Argument Coercion	252
6.6	C++ Standard Library Header Files	254
6.7	Case Study: Random Number Generation	256
6.8	Case Study: Game of Chance; Introducing enum	261
6.9	Storage Classes	265
6.10	Scope Rules	267
6.11	Function Call Stack and Activation Records	271
6.12	Functions with Empty Parameter Lists	274
6.13	Inline Functions	275
6.14	References and Reference Parameters	277
6.15	Default Arguments	281
6.16	Unary Scope Resolution Operator	283
6.17	Function Overloading	284
6.18	Function Templates	287
6.19	Recursion	289
6.20	Example Using Recursion: Fibonacci Series	292
6.21	Recursion vs. Iteration	295
6.22	Wrap-Up	298

7 Arrays and Vectors 318

7.1	Introduction	319
7.2	Arrays	320
7.3	Declaring Arrays	321
7.4	Examples Using Arrays	322
7.4.1	Declaring an Array and Using a Loop to Initialize the Array's Elements	322
7.4.2	Initializing an Array in a Declaration with an Initializer List	323
7.4.3	Specifying an Array's Size with a Constant Variable and Setting Array Elements with Calculations	324
7.4.4	Summing the Elements of an Array	327
7.4.5	Using Bar Charts to Display Array Data Graphically	327
7.4.6	Using the Elements of an Array as Counters	329
7.4.7	Using Arrays to Summarize Survey Results	330
7.4.8	Static Local Arrays and Automatic Local Arrays	333
7.5	Passing Arrays to Functions	335
7.6	Case Study: Class GradeBook Using an Array to Store Grades	339
7.7	Searching Arrays with Linear Search	345
7.8	Sorting Arrays with Insertion Sort	347
7.9	Multidimensional Arrays	349
7.10	Case Study: Class GradeBook Using a Two-Dimensional Array	352

7.11	Introduction to C++ Standard Library Class Template <code>vector</code>	359
7.12	Wrap-Up	364

8 Pointers 381

8.1	Introduction	382
8.2	Pointer Variable Declarations and Initialization	382
8.3	Pointer Operators	384
8.4	Pass-by-Reference with Pointers	386
8.5	Using <code>const</code> with Pointers	390
8.6	Selection Sort Using Pass-by-Reference	394
8.7	<code>sizeof</code> Operator	398
8.8	Pointer Expressions and Pointer Arithmetic	401
8.9	Relationship Between Pointers and Arrays	403
8.10	Pointer-Based String Processing	406
8.11	Arrays of Pointers	409
8.12	Function Pointers	410
8.13	Wrap-Up	413

9 Classes: A Deeper Look, Part 1 431

9.1	Introduction	432
9.2	Time Class Case Study	433
9.3	Class Scope and Accessing Class Members	439
9.4	Separating Interface from Implementation	441
9.5	Access Functions and Utility Functions	442
9.6	Time Class Case Study: Constructors with Default Arguments	445
9.7	Destructors	450
9.8	When Constructors and Destructors Are Called	451
9.9	Time Class Case Study: A Subtle Trap—Returning a Reference to a private Data Member	454
9.10	Default Memberwise Assignment	457
9.11	Wrap-Up	459

10 Classes: A Deeper Look, Part 2 465

10.1	Introduction	466
10.2	<code>const</code> (Constant) Objects and <code>const</code> Member Functions	466
10.3	Composition: Objects as Members of Classes	475
10.4	<code>friend</code> Functions and <code>friend</code> Classes	481
10.5	Using the <code>this</code> Pointer	484
10.6	<code>static</code> Class Members	489
10.7	Data Abstraction and Information Hiding	494
10.8	Wrap-Up	496

11 Operator Overloading 502

11.1	Introduction	503
------	--------------	-----

11.2	Fundamentals of Operator Overloading	504
11.3	Restrictions on Operator Overloading	505
11.4	Operator Functions as Class Members vs. Global Functions	506
11.5	Overloading Stream Insertion and Stream Extraction Operators	508
11.6	Overloading Unary Operators	511
11.7	Overloading Binary Operators	512
11.8	Dynamic Memory Management	512
11.9	Case Study: Array Class	514
11.10	Converting between Types	526
11.11	Building a String Class	527
11.12	Overloading ++ and --	528
11.13	Case Study: A Date Class	530
11.14	Standard Library Class string	534
11.15	explicit Constructors	538
11.16	Proxy Classes	541
11.17	Wrap-Up	545

12 Object-Oriented Programming: Inheritance 557

12.1	Introduction	558
12.2	Base Classes and Derived Classes	559
12.3	protected Members	562
12.4	Relationship between Base Classes and Derived Classes	562
12.4.1	Creating and Using a CommissionEmployee Class	563
12.4.2	Creating a BasePlusCommissionEmployee Class Without Using Inheritance	568
12.4.3	Creating a CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy	573
12.4.4	CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using protected Data	578
12.4.5	CommissionEmployee–BasePlusCommissionEmployee Inheritance Hierarchy Using private Data	585
12.5	Constructors and Destructors in Derived Classes	592
12.6	public, protected and private Inheritance	600
12.7	Software Engineering with Inheritance	601
12.8	Wrap-Up	602

13 Object-Oriented Programming: Polymorphism 608

13.1	Introduction	609
13.2	Polymorphism Examples	610
13.3	Relationships Among Objects in an Inheritance Hierarchy	611
13.3.1	Invoking Base-Class Functions from Derived-Class Objects	612
13.3.2	Aiming Derived-Class Pointers at Base-Class Objects	619
13.3.3	Derived-Class Member-Function Calls via Base-Class Pointers	620
13.3.4	Virtual Functions	622

13.3.5	Summary of the Allowed Assignments Between Base-Class and Derived-Class Objects and Pointers	628
13.4	Type Fields and switch Statements	629
13.5	Abstract Classes and Pure virtual Functions	629
13.6	Case Study: Payroll System Using Polymorphism	631
13.6.1	Creating Abstract Base Class Employee	633
13.6.2	Creating Concrete Derived Class SalariedEmployee	636
13.6.3	Creating Concrete Derived Class HourlyEmployee	638
13.6.4	Creating Concrete Derived Class CommissionEmployee	641
13.6.5	Creating Indirect Concrete Derived Class BasePlusCommissionEmployee	643
13.6.6	Demonstrating Polymorphic Processing	644
13.7	(Optional) Polymorphism, Virtual Functions and Dynamic Binding “Under the Hood”	648
13.8	Case Study: Payroll System Using Polymorphism and Runtime Type Information with Downcasting, dynamic_cast, typeid and type_info	652
13.9	Virtual Destructors	656
13.10	Wrap-Up	656

14 Templates 662

14.1	Introduction	663
14.2	Function Templates	664
14.3	Overloading Function Templates	667
14.4	Class Templates	667
14.5	Nontype Parameters and Default Types for Class Templates	674
14.6	Notes on Templates and Inheritance	675
14.7	Notes on Templates and Friends	675
14.8	Notes on Templates and static Members	676
14.9	Wrap-Up	676

15 Stream Input/Output 681

15.1	Introduction	682
15.2	Streams	683
15.2.1	Classic Streams vs. Standard Streams	683
15.2.2	iostream Library Header Files	684
15.2.3	Stream Input/Output Classes and Objects	684
15.3	Stream Output	687
15.3.1	Output of char * Variables	687
15.3.2	Character Output Using Member Function put	687
15.4	Stream Input	688
15.4.1	get and getline Member Functions	688
15.4.2	istream Member Functions peek, putback and ignore	691
15.4.3	Type-Safe I/O	691
15.5	Unformatted I/O Using read, write and gcount	691
15.6	Introduction to Stream Manipulators	692

15.6.1	Integral Stream Base: dec, oct, hex and setbase	693
15.6.2	Floating-Point Precision (precision, setprecision)	694
15.6.3	Field Width (width, setw)	695
15.6.4	User-Defined Output Stream Manipulators	696
15.7	Stream Format States and Stream Manipulators	698
15.7.1	Trailing Zeros and Decimal Points (showpoint)	698
15.7.2	Justification (left, right and internal)	699
15.7.3	Padding (fill, setfill)	701
15.7.4	Integral Stream Base (dec, oct, hex, showbase)	702
15.7.5	Floating-Point Numbers; Scientific and Fixed Notation (scientific, fixed)	703
15.7.6	Uppercase/Lowercase Control (uppercase)	704
15.7.7	Specifying Boolean Format (boolalpha)	704
15.7.8	Setting and Resetting the Format State via Member Function flags	705
15.8	Stream Error States	707
15.9	Tying an Output Stream to an Input Stream	709
15.10	Wrap-Up	709

16 Exception Handling 719

16.1	Introduction	720
16.2	Exception-Handling Overview	721
16.3	Example: Handling an Attempt to Divide by Zero	721
16.4	When to Use Exception Handling	727
16.5	Rethrowing an Exception	728
16.6	Exception Specifications	730
16.7	Processing Unexpected Exceptions	731
16.8	Stack Unwinding	731
16.9	Constructors, Destructors and Exception Handling	733
16.10	Exceptions and Inheritance	734
16.11	Processing new Failures	734
16.12	Class auto_ptr and Dynamic Memory Allocation	737
16.13	Standard Library Exception Hierarchy	739
16.14	Other Error-Handling Techniques	741
16.15	Wrap-Up	742

17 File Processing 749

17.1	Introduction	750
17.2	Data Hierarchy	750
17.3	Files and Streams	752
17.4	Creating a Sequential File	753
17.5	Reading Data from a Sequential File	757
17.6	Updating Sequential Files	762
17.7	Random-Access Files	763
17.8	Creating a Random-Access File	764

17.9	Writing Data Randomly to a Random-Access File	769
17.10	Reading from a Random-Access File Sequentially	771
17.11	Case Study: A Transaction-Processing Program	773
17.12	Overview of Object Serialization	779
17.13	Wrap-Up	780

18 Class string and String Stream Processing 791

18.1	Introduction	792
18.2	string Assignment and Concatenation	793
18.3	Comparing strings	795
18.4	Substrings	798
18.5	Swapping strings	798
18.6	string Characteristics	799
18.7	Finding Substrings and Characters in a string	802
18.8	Replacing Characters in a string	804
18.9	Inserting Characters into a string	805
18.10	Conversion to C-Style Pointer-Based char * Strings	806
18.11	Iterators	808
18.12	String Stream Processing	809
18.13	Wrap-Up	812

19 Searching and Sorting 820

19.1	Introduction	821
19.2	Searching Algorithms	822
19.2.1	Efficiency of Linear Search	822
19.2.2	Binary Search	824
19.3	Sorting Algorithms	829
19.3.1	Efficiency of Selection Sort	829
19.3.2	Efficiency of Insertion Sort	829
19.3.3	Merge Sort (A Recursive Implementation)	830
19.4	Wrap-Up	837

20 Data Structures 842

20.1	Introduction	843
20.2	Self-Referential Classes	844
20.3	Dynamic Memory Allocation and Data Structures	845
20.4	Linked Lists	845
20.5	Stacks	860
20.6	Queues	865
20.7	Trees	868
20.8	Wrap-Up	877

21 Bits, Characters, C Strings and structs 888

21.1	Introduction	889
------	--------------	-----

21.2	Structure Definitions	889
21.3	Initializing Structures	892
21.4	Using Structures with Functions	892
21.5	typedef	892
21.6	Example: Card Shuffling and Dealing Simulation	893
21.7	Bitwise Operators	896
21.8	Bit Fields	905
21.9	Character-Handling Library	909
21.10	Pointer-Based String Manipulation Functions	914
21.11	Pointer-Based String-Conversion Functions	921
21.12	Search Functions of the Pointer-Based String-Handling Library	926
21.13	Memory Functions of the Pointer-Based String-Handling Library	931
21.14	Wrap-Up	935

22 Standard Template Library (STL) 952

22.1	Introduction to the Standard Template Library (STL)	953
22.1.1	Introduction to Containers	955
22.1.2	Introduction to Iterators	959
22.1.3	Introduction to Algorithms	964
22.2	Sequence Containers	966
22.2.1	vector Sequence Container	966
22.2.2	list Sequence Container	974
22.2.3	deque Sequence Container	978
22.3	Associative Containers	980
22.3.1	multiset Associative Container	980
22.3.2	set Associative Container	983
22.3.3	multimap Associative Container	984
22.3.4	map Associative Container	986
22.4	Container Adapters	988
22.4.1	stack Adapter	988
22.4.2	queue Adapter	990
22.4.3	priority_queue Adapter	991
22.5	Algorithms	993
22.5.1	fill, fill_n, generate and generate_n	994
22.5.2	equal, mismatch and lexicographical_compare	995
22.5.3	remove, remove_if, remove_copy and remove_copy_if	998
22.5.4	replace, replace_if, replace_copy and replace_copy_if	1000
22.5.5	Mathematical Algorithms	1003
22.5.6	Basic Searching and Sorting Algorithms	1006
22.5.7	swap, iter_swap and swap_ranges	1008
22.5.8	copy_backward, merge, unique and reverse	1009
22.5.9	inplace_merge, unique_copy and reverse_copy	1012
22.5.10	Set Operations	1013
22.5.11	lower_bound, upper_bound and equal_range	1016
22.5.12	Heapsort	1018

22.5.13	min and max	1021
22.5.14	STL Algorithms Not Covered in This Chapter	1022
22.6	Class bitset	1023
22.7	Function Objects	1027
22.8	Wrap-Up	1030
22.9	STL Web Resources	1031

Chapters on the Web

1041

Chapters 23–27 are PDF documents posted online at the book’s Companion Website (located at www.pearsonhighered.com/deitel).

23 Boost Libraries, Technical Report 1 and C++0x I

23.1	Introduction	II
23.2	Deitel Online C++ and Related Resource Centers	II
23.3	Boost Libraries	II
23.4	Boost Libraries Overview	III
23.5	Regular Expressions with the Boost.Regex Library	VI
23.5.1	Regular Expression Example	VI
23.5.2	Validating User Input with Regular Expressions	IX
23.5.3	Replacing and Splitting Strings	XII
23.6	Smart Pointers with Boost.Smart_ptr	XIV
23.6.1	Reference Counted shared_ptr	XIV
23.6.2	weak_ptr: shared_ptr Observer	XIX
23.7	Technical Report 1	XXIV
23.8	C++0x	XXVI
23.9	Core Language Changes	XXVI
23.10	Wrap-Up	XXXI

24 Other Topics XL

24.1	Introduction	XLI
24.2	const_cast Operator	XLI
24.3	mutable Class Members	XLIII
24.4	namespaces	XLV
24.5	Operator Keywords	XLVIII
24.6	Pointers to Class Members (. * and ->*)	L
24.7	Multiple Inheritance	LII
24.8	Multiple Inheritance and virtual Base Classes	LVII
24.9	Wrap-Up	LXII

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

LXVII

25.1	Introduction	LXVIII
25.2	Examining the ATM Requirements Document	LXVIII

25.3	Identifying the Classes in the ATM Requirements Document	LXXVI
25.4	Identifying Class Attributes	LXXXIII
25.5	Identifying Objects' States and Activities	LXXXVII
25.6	Identifying Class Operations	XCI
25.7	Indicating Collaboration Among Objects	XCVIII
25.8	Wrap-Up	CV

26 ATM Case Study, Part 2: Implementing an Object-Oriented Design CIX

26.1	Introduction	CX
26.2	Starting to Program the Classes of the ATM System	CX
26.3	Incorporating Inheritance into the ATM System	CXVII
26.4	ATM Case Study Implementation	CXXIV
26.4.1	Class ATM	CXXIV
26.4.2	Class Screen	CXXXII
26.4.3	Class Keypad	CXXXIII
26.4.4	Class CashDispenser	CXXXIV
26.4.5	Class DepositSlot	CXXXVI
26.4.6	Class Account	CXXXVII
26.4.7	Class BankDatabase	CXXXIX
26.4.8	Class Transaction	CXLIII
26.4.9	Class BalanceInquiry	CXLV
26.4.10	Class Withdrawal	CXLVII
26.4.11	Class Deposit	CLII
26.4.12	Test Program ATMCaseStudy.cpp	CLV
26.5	Wrap-Up	CLV

27 Game Programming with Ogre CLVIII

27.1	Introduction	CLIX
27.2	Installing Ogre, OgreAL and OpenAL	CLIX
27.3	Basics of Game Programming	CLIX
27.4	The Game of Pong: Code Walkthrough	CLXII
27.4.1	Ogre Initialization	CLXIII
27.4.2	Creating a Scene	CLXXII
27.4.3	Adding to the Scene	CLXXIII
27.4.4	Animation and Timers	CLXXXV
27.4.5	User Input	CLXXXVI
27.4.6	Collision Detection	CLXXXVIII
27.4.7	Sound	CXCII
27.4.8	Resources	CXCIII
27.4.9	Pong Driver	CXCIV
27.5	Wrap-Up	CXCV
27.6	Ogre Web Resources	CXCV

A	Operator Precedence and Associativity	1042
B	ASCII Character Set	1044
C	Fundamental Types	1045
D	Number Systems	1047
D.1	Introduction	1048
D.2	Abbreviating Binary Numbers as Octal and Hexadecimal Numbers	1051
D.3	Converting Octal and Hexadecimal Numbers to Binary Numbers	1052
D.4	Converting from Binary, Octal or Hexadecimal to Decimal	1052
D.5	Converting from Decimal to Binary, Octal or Hexadecimal	1053
D.6	Negative Binary Numbers: Two's Complement Notation	1055
E	Preprocessor	1060
E.1	Introduction	1061
E.2	<code>#include</code> Preprocessor Directive	1061
E.3	<code>#define</code> Preprocessor Directive: Symbolic Constants	1062
E.4	<code>#define</code> Preprocessor Directive: Macros	1062
E.5	Conditional Compilation	1064
E.6	<code>#error</code> and <code>#pragma</code> Preprocessor Directives	1065
E.7	Operators <code>#</code> and <code>##</code>	1066
E.8	Predefined Symbolic Constants	1066
E.9	Assertions	1067
E.10	Wrap-Up	1067
	Appendices on the Web	1072
<p>Appendices F–I are PDF documents posted online at the book's Companion Website (located at www.pearsonhighered.com/deitel).</p>		
F	C Legacy Code Topics	CCV
F.1	Introduction	CCVI
F.2	Redirecting Input/Output on UNIX/Linux/Mac OS X and Windows Systems	CCVI
F.3	Variable-Length Argument Lists	CCVII
F.4	Using Command-Line Arguments	CCIX
F.5	Notes on Compiling Multiple-Source-File Programs	CCXI
F.6	Program Termination with <code>exit</code> and <code>atexit</code>	CCXIII
F.7	Type Qualifier <code>volatile</code>	CCXIV
F.8	Suffixes for Integer and Floating-Point Constants	CCXIV
F.9	Signal Handling	CCXV
F.10	Dynamic Memory Allocation with <code>calloc</code> and <code>realloc</code>	CCXVII

F.11	Unconditional Branch: goto	CCXVIII
F.12	Unions	CCXIX
F.13	Linkage Specifications	CCXXII
F.14	Wrap-Up	CCXXIII

G UML 2: Additional Diagram Types CCXXIX

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

H Using the Visual Studio Debugger CCXXXI

H.1	Introduction	CCXXXII
H.2	Breakpoints and the Continue Command	CCXXXII
H.3	Locals and Watch Windows	CCXXXVIII
H.4	Controlling Execution Using the Step Into, Step Over, Step Out and Continue Commands	CCXLI
H.5	Autos Window	CCXLIII
H.6	Wrap-Up	CCXLIV

I Using the GNU C++ Debugger CCXLVII

I.1	Introduction	CCXLVIII
I.2	Breakpoints and the run, stop, continue and print Commands	CCXLVIII
I.3	print and set Commands	CCLIV
I.4	Controlling Execution Using the step, finish and next Commands	CCLVI
I.5	watch Command	CCLIX
I.6	Wrap-Up	CCLXI

Index 1073