

C++ How to Program, 7/E
Paul Deitel; Harvey M. Deitel
ISBN: 9780136117261

Table of Contents

Chapters 23—27 and Appendices F—I are PDF documents posted online at the book's Companion Website (located at www.pearsonhighered.com/deitel).

Preface xxiii

1	Introduction to Computers, the Internet and the World Wide Web	1
1.1	Introduction	2
1.2	Computers: Hardware and Software	3
1.3	Computer Organization	4
1.4	Personal, Distributed and Client/Server Computing	5
1.5	The Internet and the World Wide Web	6
1.6	Web 2.0	6
1.7	Machine Languages, Assembly Languages and High-Level Languages	7
1.8	History of C and C++	8
1.9	C++ Standard Library	9
1.10	History of Java	10
1.11	Fortran, COBOL, Pascal and Ada	11
1.12	BASIC, Visual Basic, Visual C++, C# and .NET	11
1.13	Key Software Trend: Object Technology	12
1.14	Typical C++ Development Environment	13
1.15	Notes About C++ and C++ How to Program, 7/e	15
1.16	Test-Driving a C++ Application	16
1.17	Software Technologies	22
1.18	Future of C++: Open Source Boost Libraries, TR1 and C++0x	23
1.19	Software Engineering Case Study: Introduction to Object Technology and the UML	24
1.20	Wrap-Up	28
1.21	Web Resources	29
2	Introduction to C++ Programming	39
2.1	Introduction	40
2.2	First Program in C++: Printing a Line of Text	40
2.3	Modifying Our First C++ Program	44
2.4	Another C++ Program: Adding Integers	45

2.5	Memory Concepts	49
2.6	Arithmetic	50
2.7	Decision Making: Equality and Relational Operators	54
2.8	Wrap-Up	58
3	Introduction to Classes and Objects	68
3.1	Introduction	69
3.2	Classes, Objects, Member Functions and Data Members	69
3.3	Defining a Class with a Member Function	71
3.4	Defining a Member Function with a Parameter	74
3.5	Data Members, set Functions and get Functions	77
3.6	Initializing Objects with Constructors	84
3.7	Placing a Class in a Separate File for Reusability	87
3.8	Separating Interface from Implementation	91
3.9	Validating Data with set Functions	97
3.10	Wrap-Up	102
4	Control Statements: Part 1	109
4.1	Introduction	110
4.2	Algorithms	110
4.3	Pseudocode	111
4.4	Control Structures	112
4.5	if Selection Statement	115
4.6	if...else Double-Selection Statement	117
4.7	while Repetition Statement	122
4.8	Formulating Algorithms: Counter-Controlled Repetition	123
4.9	Formulating Algorithms: Sentinel-Controlled Repetition	129
4.10	Formulating Algorithms: Nested Control Statements	139
4.11	Assignment Operators	144
4.12	Increment and Decrement Operators	144
4.13	Wrap-Up	148
5	Control Statements: Part 2	163
5.1	Introduction	164
5.2	Essentials of Counter-Controlled Repetition	164
5.3	for Repetition Statement	166
5.4	Examples Using the for Statement	170
5.5	do...while Repetition Statement	174

5.6	switch Multiple-Selection Statement	176
5.7	break and continue Statements	185
5.8	Logical Operators	187
5.9	Confusing the Equality (==) and Assignment (=) Operators	191
5.10	Structured Programming Summary	192
5.11	Wrap-Up	197
6	Functions and an Introduction to Recursion	207
6.1	Introduction	208
6.2	Program Components in C++	209
6.3	Math Library Functions	210
6.4	Function Definitions with Multiple Parameters	211
6.5	Function Prototypes and Argument Coercion	216
6.6	C++ Standard Library Header Files	218
6.7	Case Study: Random Number Generation	220
6.8	Case Study: Game of Chance; Introducing enum	225
6.9	Storage Classes	229
6.10	Scope Rules	231
6.11	Function Call Stack and Activation Records	235
6.12	Functions with Empty Parameter Lists	238
6.13	Inline Functions	239
6.14	References and Reference Parameters	241
6.15	Default Arguments	245
6.16	Unary Scope Resolution Operator	247
6.17	Function Overloading	248
6.18	Function Templates	251
6.19	Recursion	253
6.20	Example Using Recursion: Fibonacci Series	256
6.21	Recursion vs. Iteration	259
6.22	Wrap-Up	262
7	Arrays and Vectors	282
7.1	Introduction	283
7.2	Arrays	284
7.3	Declaring Arrays	285
7.4	Examples Using Arrays	286
7.4.1	Declaring an Array and Using a Loop to Initialize the Array's Elements	286

7.4.2	Initializing an Array in a Declaration with an Initializer List	287
7.4.3	Specifying an Array's Size with a Constant Variable and Setting Array Elements with Calculations	288
7.4.4	Summing the Elements of an Array	291
7.4.5	Using Bar Charts to Display Array Data Graphically	291
7.4.6	Using the Elements of an Array as Counters	293
7.4.7	Using Arrays to Summarize Survey Results	294
7.4.8	Static Local Arrays and Automatic Local Arrays	297
7.5	Passing Arrays to Functions	299
7.6	Case Study: Class GradeBook Using an Array to Store Grades	303
7.7	Searching Arrays with Linear Search	309
7.8	Sorting Arrays with Insertion Sort	311
7.9	Multidimensional Arrays	313
7.10	Case Study: Class GradeBook Using a Two-Dimensional Array	316
7.11	Introduction to C++ Standard Library Class Template vector	323
7.12	Wrap-Up	328
8	Pointers	345
8.1	Introduction	346
8.2	Pointer Variable Declarations and Initialization	346
8.3	Pointer Operators	348
8.4	Pass-by-Reference with Pointers	350
8.5	Using const with Pointers	354
8.6	Selection Sort Using Pass-by-Reference	358
8.7	sizeof Operator	362
8.8	Pointer Expressions and Pointer Arithmetic	365
8.9	Relationship Between Pointers and Arrays	367
8.10	Pointer-Based String Processing	370
8.11	Arrays of Pointers	373
8.12	Function Pointers	374
8.13	Wrap-Up	377
9	Classes: A Deeper Look, Part 1	395
9.1	Introduction	396
9.2	Time Class Case Study	397
9.3	Class Scope and Accessing Class Members	403
9.4	Separating Interface from Implementation	405

9.5	Access Functions and Utility Functions	406
9.6	Time Class Case Study: Constructors with Default Arguments	409
9.7	Destructors	414
9.8	When Constructors and Destructors Are Called	415
9.9	Time Class Case Study: A Subtle Trap—Returning a Reference to a private Data Member	418
9.10	Default Memberwise Assignment	421
9.11	Wrap-Up	423
10	Classes: A Deeper Look, Part 2	429
10.1	Introduction	430
10.2	const (Constant) Objects and const Member Functions	430
10.3	Composition: Objects as Members of Classes	439
10.4	friend Functions and friend Classes	445
10.5	Using the this Pointer	448
10.6	static Class Members	453
10.7	Data Abstraction and Information Hiding	458
10.8	Wrap-Up	460
11	Operator Overloading	466
11.1	Introduction	467
11.2	Fundamentals of Operator Overloading	468
11.3	Restrictions on Operator Overloading	469
11.4	Operator Functions as Class Members vs. Global Functions	470
11.5	Overloading Stream Insertion and Stream Extraction Operators	472
11.6	Overloading Unary Operators	475
11.7	Overloading Binary Operators	476
11.8	Dynamic Memory Management	476
11.9	Case Study: Array Class	478
11.10	Converting between Types	490
11.11	Building a String Class	491
11.12	Overloading ++ and --	492
11.13	Case Study: A Date Class	494
11.14	Standard Library Class string	498
11.15	explicit Constructors	502
11.16	Proxy Classes	505
11.17	Wrap-Up	509

12 Object-Oriented Programming: Inheritance	521
12.1 Introduction	522
12.2 Base Classes and Derived Classes	523
12.3 protected Members	526
12.4 Relationship between Base Classes and Derived Classes	526
12.4.1 Creating and Using a CommissionEmployee Class	527
12.4.2 Creating a BasePlusCommissionEmployee Class Without Using Inheritance	532
12.4.3 Creating a CommissionEmployee—BasePlusCommissionEmployee Inheritance Hierarchy	537
12.4.4 CommissionEmployee—BasePlusCommissionEmployee Inheritance Hierarchy Using protected Data	542
12.4.5 CommissionEmployee—BasePlusCommissionEmployee Inheritance Hierarchy Using private Data	549
12.5 Constructors and Destructors in Derived Classes	556
12.6 public, protected and private Inheritance	564
12.7 Software Engineering with Inheritance	565
12.8 Wrap-Up	566
13 Object-Oriented Programming: Polymorphism	572
13.1 Introduction	573
13.2 Polymorphism Examples	574
13.3 Relationships Among Objects in an Inheritance Hierarchy	575
13.3.1 Invoking Base-Class Functions from Derived-Class Objects	576
13.3.2 Aiming Derived-Class Pointers at Base-Class Objects	583
13.3.3 Derived-Class Member-Function Calls via Base-Class Pointers	584
13.3.4 Virtual Functions	586
13.3.5 Summary of the Allowed Assignments Between Base-Class and Derived-Class Objects and Pointers	592
13.4 Type Fields and switch Statements	593
13.5 Abstract Classes and Pure virtual Functions	593
13.6 Case Study: Payroll System Using Polymorphism	595
13.6.1 Creating Abstract Base Class Employee	597
13.6.2 Creating Concrete Derived Class SalariedEmployee	600
13.6.3 Creating Concrete Derived Class HourlyEmployee	602
13.6.4 Creating Concrete Derived Class CommissionEmployee	605
13.6.5 Creating Indirect Concrete Derived Class BasePlusCommissionEmployee	607

13.6.6 Demonstrating Polymorphic Processing	608
13.7 (Optional) Polymorphism, Virtual Functions and Dynamic Binding “Under the Hood”	612
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>	616
13.9 Virtual Destructors	620
13.10 Wrap-Up	620
14 Templates	626
14.1 Introduction	627
14.2 Function Templates	628
14.3 Overloading Function Templates	631
14.4 Class Templates	631
14.5 Nontype Parameters and Default Types for Class Templates	638
14.6 Notes on Templates and Inheritance	639
14.7 Notes on Templates and Friends	639
14.8 Notes on Templates and static Members	640
14.9 Wrap-Up	640
15 Stream Input/Output	645
15.1 Introduction	646
15.2 Streams	647
15.2.1 Classic Streams vs. Standard Streams	647
15.2.2 <code>iostream</code> Library Header Files	648
15.2.3 Stream Input/Output Classes and Objects	648
15.3 Stream Output	651
15.3.1 Output of <code>char *</code> Variables	651
15.3.2 Character Output Using Member Function <code>put</code>	651
15.4 Stream Input	652
15.4.1 <code>get</code> and <code>getline</code> Member Functions	652
15.4.2 <code>istream</code> Member Functions <code>peek</code> , <code>putback</code> and <code>ignore</code>	655
15.4.3 Type-Safe I/O	655
15.5 Unformatted I/O Using <code>read</code> , <code>write</code> and <code>gcount</code>	655
15.6 Introduction to Stream Manipulators	656
15.6.1 Integral Stream Base: <code>dec</code> , <code>oct</code> , <code>hex</code> and <code>setbase</code>	657
15.6.2 Floating-Point Precision (<code>precision</code> , <code>setprecision</code>)	658
15.6.3 Field Width (<code>width</code> , <code>setw</code>)	659

15.6.4	User-Defined Output Stream Manipulators	660
15.7	Stream Format States and Stream Manipulators	662
15.7.1	Trailing Zeros and Decimal Points (showpoint)	662
15.7.2	Justification (left, right and internal)	663
15.7.3	Padding (fill, setfill)	665
15.7.4	Integral Stream Base (dec, oct, hex, showbase)	666
15.7.5	Floating-Point Numbers; Scientific and Fixed Notation (scientific, fixed)	667
15.7.6	Uppercase/Lowercase Control (uppercase)	668
15.7.7	Specifying Boolean Format (boolalpha)	668
15.7.8	Setting and Resetting the Format State via Member Function flags	669
15.8	Stream Error States	671
15.9	Tying an Output Stream to an Input Stream	673
15.10	Wrap-Up	673
16	Exception Handling	683
16.1	Introduction	684
16.2	Exception-Handling Overview	685
16.3	Example: Handling an Attempt to Divide by Zero	685
16.4	When to Use Exception Handling	691
16.5	Rethrowing an Exception	692
16.6	Exception Specifications	694
16.7	Processing Unexpected Exceptions	695
16.8	Stack Unwinding	695
16.9	Constructors, Destructors and Exception Handling	697
16.10	Exceptions and Inheritance	698
16.11	Processing new Failures	698
16.12	Class auto_ptr and Dynamic Memory Allocation	701
16.13	Standard Library Exception Hierarchy	703
16.14	Other Error-Handling Techniques	705
16.15	Wrap-Up	706
17	File Processing	713
17.1	Introduction	714
17.2	Data Hierarchy	714
17.3	Files and Streams	716
17.4	Creating a Sequential File	717
17.5	Reading Data from a Sequential File	721

17.6	Updating Sequential Files	726
17.7	Random-Access Files	727
17.8	Creating a Random-Access File	728
17.9	Writing Data Randomly to a Random-Access File	733
17.10	Reading from a Random-Access File Sequentially	735
17.11	Case Study: A Transaction-Processing Program	737
17.12	Overview of Object Serialization	743
17.13	Wrap-Up	744
18	Class string and String Stream Processing	755
18.1	Introduction	756
18.2	string Assignment and Concatenation	757
18.3	Comparing strings	759
18.4	Substrings	762
18.5	Swapping strings	762
18.6	string Characteristics	763
18.7	Finding Substrings and Characters in a string	766
18.8	Replacing Characters in a string	768
18.9	Inserting Characters into a string	769
18.10	Conversion to C-Style Pointer-Based char * Strings	770
18.11	Iterators	772
18.12	String Stream Processing	773
18.13	Wrap-Up	776
19	Searching and Sorting	784
19.1	Introduction	785
19.2	Searching Algorithms	786
19.2.1	Efficiency of Linear Search	786
19.2.2	Binary Search	788
19.3	Sorting Algorithms	793
19.3.1	Efficiency of Selection Sort	793
19.3.2	Efficiency of Insertion Sort	793
19.3.3	Merge Sort (A Recursive Implementation)	794
19.4	Wrap-Up	801
20	Data Structures	806
20.1	Introduction	807
20.2	Self-Referential Classes	808

20.3	Dynamic Memory Allocation and Data Structures	809
20.4	Linked Lists	809
20.5	Stacks	824
20.6	Queues	829
20.7	Trees	832
20.8	Wrap-Up	841
21	Bits, Characters, C Strings and structs	852
21.1	Introduction	853
21.2	Structure Definitions	853
21.3	Initializing Structures	856
21.4	Using Structures with Functions	856
21.5	typedef	856
21.6	Example: Card Shuffling and Dealing Simulation	857
21.7	Bitwise Operators	860
21.8	Bit Fields	869
21.9	Character-Handling Library	873
21.10	Pointer-Based String Manipulation Functions	878
21.11	Pointer-Based String-Conversion Functions	885
21.12	Search Functions of the Pointer-Based String-Handling Library	890
21.13	Memory Functions of the Pointer-Based String-Handling Library	895
21.14	Wrap-Up	899
22	Standard Template Library (STL)	916
22.1	Introduction to the Standard Template Library (STL)	917
22.1.1	Introduction to Containers	919
22.1.2	Introduction to Iterators	923
22.1.3	Introduction to Algorithms	928
22.2	Sequence Containers	930
22.2.1	vector Sequence Container	930
22.2.2	list Sequence Container	938
22.2.3	deque Sequence Container	942
22.3	Associative Containers	944
22.3.1	multiset Associative Container	944
22.3.2	set Associative Container	947
22.3.3	multimap Associative Container	948
22.3.4	map Associative Container	950

22.4 Container Adapters	952
22.4.1 stack Adapter	952
22.4.2 queue Adapter	954
22.4.3 priority_queue Adapter	955
22.5 Algorithms	957
22.5.1 fill, fill_n, generate and generate_n	958
22.5.2 equal, mismatch and lexicographical_compare	959
22.5.3 remove, remove_if, remove_copy and remove_copy_if	962
22.5.4 replace, replace_if, replace_copy and replace_copy_if	964
22.5.5 Mathematical Algorithms	967
22.5.6 Basic Searching and Sorting Algorithms	970
22.5.7 swap, iter_swap and swap_ranges	972
22.5.8 copy_backward, merge, unique and reverse	973
22.5.9 inplace_merge, unique_copy and reverse_copy	976
22.5.10 Set Operations	977
22.5.11 lower_bound, upper_bound and equal_range	980
22.5.12 Heapsort	982
22.5.13 min and max	985
22.5.14 STL Algorithms Not Covered in This Chapter	986
22.6 Class bitset	987
22.7 Function Objects	991
22.8 Wrap-Up	994
22.9 STL Web Resources	995
Chapters on the Web	1005
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 1: 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 CXCV
- 27.5 Wrap-Up CXCV
- 27.6 Ogre Web Resources CXCV

- A Operator Precedence and Associativity 1006
- B ASCII Character Set 1008
- C Fundamental Types 1009
- D Number Systems 1011
- D.1 Introduction 1012
- D.2 Abbreviating Binary Numbers as Octal and Hexadecimal Numbers 1015
- D.3 Converting Octal and Hexadecimal Numbers to Binary Numbers 1016
- D.4 Converting from Binary, Octal or Hexadecimal to Decimal 1016
- D.5 Converting from Decimal to Binary, Octal or Hexadecimal 1017
- D.6 Negative Binary Numbers: Two's Complement Notation 1019
- E Preprocessor 1024
- E.1 Introduction 1025

- E.2 #include Preprocessor Directive 1025
- E.3 #define Preprocessor Directive: Symbolic Constants 1026
- E.4 #define Preprocessor Directive: Macros 1026
- E.5 Conditional Compilation 1028
- E.6 #error and #pragma Preprocessor Directives 1029
- E.7 Operators # and ## 1030
- E.8 Predefined Symbolic Constants 1030
- E.9 Assertions 1031
- E.10 Wrap-Up 1031
- Appendices on the Web 1036

Appendices F—I are PDF documents posted online at the book's Companion Website (located at www.pearsonhighered.com/deitel).

- 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 exit and atexit CCXIII
 - F.7 Type Qualifier volatile CCXIV
 - F.8 Suffixes for Integer and Floating-Point Constants CCXIV
 - F.9 Signal Handling CCXV
 - F.10 Dynamic Memory Allocation with calloc and realloc 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 1037