
Contents

Preface	xxiv
7 Introduction to Computers and C++ Programming	1
1.1 Introduction	2
1.2 What is a Computer?	5
1.3 Computer Organization	5
1.4 Evolution of Operating Systems	6
1.5 Personal Computing, Distributed Computing and Client/Server Computing	7
1.6 Machine Languages, Assembly Languages, and High-level Languages	8
1.7 History of C and C++	9
1.8 C++ Standard Library	10
1.9 Java and <i>Java How to Program</i>	11
1.10 Other High-level Languages	11
1.11 Structured Programming	12
1.12 The Key Software Trend: Object Technology	12
1.13 Basics of a Typical C++ Environment	15
1.14 Hardware Trends	17
1.15 History of the Internet	18
1.16 History of the World Wide Web	19
1.17 General Notes About C++ and This Book	19
1.18 Introduction to C++ Programming	20
1.19 A Simple Program: Printing a Line of Text	21
1.20 Another Simple Program: Adding Two Integers	25
1.21 Memory Concepts	29
1.22 Arithmetic	30
1.23 Decision Making: Equality and Relational Operators	34
1.24 Thinking About Objects: Introduction to Object Technology and the Unified Modeling Language™	38

©1994–2000 by Deitel & Associates, Inc. and Prentice Hall. All rights reserved.

2	Control Structures	58
2.1	Introduction	59
2.2	Algorithms	60
2.3	Pseudocode	60
2.4	Control Structures	61
2.5	The if Selection Structure	63
2.6	The if/else Selection Structure	65
2.7	The while Repetition Structure	69
2.8	Formulating Algorithms: Case Study 1 (Counter-Controlled Repetition)	70
2.9	Formulating Algorithms with Top-Down, Stepwise Refinement: Case Study 2 (Sentinel-Controlled Repetition)	73
2.10	Formulating Algorithms with Top-Down, Stepwise Refinement: Case Study 3 (Nested Control Structures)	81
2.11	Assignment Operators	85
2.12	Increment and Decrement Operators	86
2.13	Essentials of Counter-Controlled Repetition	88
2.14	The for Repetition Structure	91
2.15	Examples Using the for Structure	95
2.16	The switch Multiple-Selection Structure	99
2.17	The do/while Repetition Structure	105
2.18	The break and continue Statements	107
2.19	Logical Operators	109
2.20	Confusing Equality (==) and Assignment (=) Operators	113
2.21	Structured-Programming Summary	114
2.22	[Optional Case Study] Thinking About Objects: Identifying the Classes in a Problem	119
3	Functions	157
3.1	Introduction	158
3.2	Program Components in C++	158
3.3	Math Library Functions	159
3.4	Functions	160
3.5	Function Definitions	162
3.6	Function Prototypes	166
3.7	Header Files	168
3.8	Random Number Generation	170
3.9	Example: A Game of Chance and Introducing enum	176
3.10	Storage Classes	179
3.11	Scope Rules	182
3.12	Recursion	185
3.13	Example Using Recursion: The Fibonacci Series	188
3.14	Recursion vs. Iteration	192
3.15	Functions with Empty Parameter Lists	194
3.16	Inline Functions	195
3.17	References and Reference Parameters	196

3.18	Default Arguments	201
3.19	Unary Scope Resolution Operator	202
3.20	Function Overloading	203
3.21	Function Templates	206
3.22	[Optional Case Study] Thinking About Objects: Identifying a Class's Attributes	208
4	Arrays	239
4.1	Introduction	240
4.2	Arrays	240
4.3	Declaring Arrays	242
4.4	Examples Using Arrays	242
4.5	Passing Arrays to Functions	258
4.6	Sorting Arrays	262
4.7	Case Study: Computing Mean, Median and Mode Using Arrays	264
4.8	Searching Arrays: Linear Search and Binary Search	269
4.9	Multiple-Subscripted Arrays	273
4.10	[Optional Case Study] Thinking About Objects: Identifying the Operations of a Class	280
5	Pointers and Strings	304
5.1	Introduction	305
5.2	Pointer Variable Declarations and Initialization	306
5.3	Pointer Operators	307
5.4	Calling Functions by Reference	310
5.5	Using the const Qualifier with Pointers	314
5.6	Bubble Sort Using Call-by-reference	321
5.7	Pointer Expressions and Pointer Arithmetic	326
5.8	The Relationship Between Pointers and Arrays	328
5.9	Arrays of Pointers	333
5.10	Case Study: A Card Shuffling and Dealing Simulation	333
5.11	Function Pointers	338
5.12	Introduction to Character and String Processing	343
	5.12.1 Fundamentals of Characters and Strings	343
	5.12.2 String Manipulation Functions of the String-handling Library	345
5.13	[Optional Case Study] Thinking About Objects: Collaborations Among Objects	353
6	Classes and Data Abstraction	389
6.1	Introduction	390
6.2	Structure Definitions	391
6.3	Accessing Members of Structures	392
6.4	Implementing a User-Defined Type Time with a struct	393
6.5	Implementing a Time Abstract Data Type with a class	395
6.6	Class Scope and Accessing Class Members	402
6.7	Separating Interface from Implementation	404
6.8	Controlling Access to Members	407

6.9	Access Functions and Utility Functions	410
6.10	Initializing Class Objects: Constructors	413
6.11	Using Default Arguments with Constructors	414
6.12	Using Destructors	418
6.13	When Constructors and Destructors Are Called	418
6.14	Using Data Members and Member Functions	421
6.15	A Subtle Trap: Returning a Reference to a Private Data Member	426
6.16	Assignment by Default Memberwise Copy	429
6.17	Software Reusability	430
6.18	[Optional Case Study] Thinking About Objects: Starting to Program the Classes for the Elevator Simulator	431
7	Classes: Part II	452
7.1	Introduction	453
7.2	const (Constant) Objects and const Member Functions	453
7.3	Composition: Objects as Members of Classes	462
7.4	friend Functions and friend Classes	467
7.5	Using the this Pointer	471
7.6	Dynamic Memory Allocation with Operators new and delete	476
7.7	static Class Members	477
7.8	Data Abstraction and Information Hiding	483
7.8.1	Example: Array Abstract Data Type	484
7.8.2	Example: String Abstract Data Type	485
7.8.3	Example: Queue Abstract Data Type	485
7.9	Container Classes and Iterators	486
7.10	Proxy Classes	486
7.10	[Optional Case Study] Thinking About Objects: Programming the Classes for the Elevator Simulator	488
8	Operator Overloading	523
8.1	Introduction	524
8.2	Fundamentals of Operator Overloading	525
8.3	Restrictions on Operator Overloading	526
8.4	Operator Functions as Class Members vs. as friend Functions	528
8.5	Overloading Stream-Insertion and Stream-Extraction Operators	529
8.6	Overloading Unary Operators	532
8.7	Overloading Binary Operators	532
8.8	Case Study: An Array Class	533
8.9	Converting between Types	545
8.10	Case Study: A String Class	546
8.11	Overloading ++ and --	558
8.12	Case Study: A Date Class	559
9	Inheritance	576
9.1	Introduction	577
9.2	Inheritance: Base Classes and Derived Classes	579
9.3	Protected Members	581

9.4	Casting Base-Class Pointers to Derived-Class Pointers	581
9.5	Using Member Functions	587
9.6	Overriding Base-Class Members in a Derived Class	587
9.7	public , protected and private Inheritance	592
9.8	Direct Base Classes and Indirect Base Classes	593
9.9	Using Constructors and Destructors in Derived Classes	593
9.10	Implicit Derived-Class Object to Base-Class Object Conversion	597
9.11	Software Engineering with Inheritance	598
9.12	Composition vs. Inheritance	599
9.13	“Uses A” and “Knows A” Relationships	600
9.14	Case Study: Point, Circle, Cylinder	600
9.15	Multiple Inheritance	607
9.16	[Optional Case Study] Thinking About Objects: Incorporating Inheritance into the Elevator Simulation	612
10	Virtual Functions and Polymorphism	625
10.1	Introduction	626
10.2	Type Fields and switch Statements	626
10.3	virtual Functions	627
10.4	Abstract Base Classes and Concrete Classes	628
10.5	Polymorphism	628
10.6	Case Study: A Payroll System Using Polymorphism	630
10.7	New Classes and Dynamic Binding	642
10.8	virtual Destructors	642
10.9	Case Study: Inheriting Interface and Implementation	643
10.10	Polymorphism, virtual Functions and Dynamic Binding “Under the Hood”	651
11	C++ Stream Input/Output	659
11.1	Introduction	661
11.2	Streams	661
	11.2.1 Iostream Library Header Files	662
	11.2.2 Stream Input/Output Classes and Objects	662
11.3	Stream Output	664
	11.3.1 Stream-Insertion Operator	664
	11.3.2 Cascading Stream-Insertion/Extraction Operators	666
	11.3.3 Output of char * Variables	667
	11.3.4 Character Output with Member Function put ; Cascading puts	668
11.4	Stream Input	668
	11.4.1 Stream-Extraction Operator	668
	11.4.2 get and getline Member Functions	671
	11.4.3 istream Member Functions peek , putback and ignore	674
	11.4.4 Type-Safe I/O	674
11.5	Unformatted I/O with read, gcount and write	674
11.6	Stream Manipulators	675
	11.6.1 Integral Stream Base: dec , oct , hex and setbase	675
	11.6.2 Floating-Point Precision (precision , setprecision)	676

11.6.3	Field Width (setw , width)	678
11.6.4	User-Defined Manipulators	679
11.7	Stream Format States	680
11.7.1	Format State Flags	680
11.7.2	Trailing Zeros and Decimal Points (ios::showpoint)	681
11.7.3	Justification (ios::left , ios::right , ios::internal)	682
11.7.4	Padding (fill , setfill)	684
11.7.5	Integral Stream Base (ios::dec , ios::oct , ios::hex , ios::showbase)	686
11.7.6	Floating-Point Numbers; Scientific Notation (ios::scientific , ios::fixed)	687
11.7.7	Uppercase/Lowercase Control (ios::uppercase)	688
11.7.8	Setting and Resetting the Format Flags (flags , setiosflags , resetiosflags)	688
11.8	Stream Error States	690
11.9	Tying an Output Stream to an Input Stream	692
12	Templates	704
12.1	Introduction	705
12.2	Function Templates	706
12.3	Overloading Template Functions	709
12.4	Class Templates	709
12.5	Class Templates and Nontype Parameters	715
12.6	Templates and Inheritance	716
12.7	Templates and friends	716
12.8	Templates and static Members	717
13	Exception Handling	723
13.1	Introduction	724
13.2	When Exception Handling Should Be Used	726
13.3	Other Error-Handling Techniques	727
13.4	Basics of C++ Exception Handling: try , throw , catch	728
13.5	A Simple Exception-Handling Example: Divide by Zero	728
13.6	Throwing an Exception	731
13.7	Catching an Exception	732
13.8	Rethrowing an Exception	735
13.9	Exception Specifications	737
13.10	Processing Unexpected Exceptions	737
13.11	Stack Unwinding	738
13.12	Constructors, Destructors and Exception Handling	739
13.13	Exceptions and Inheritance	740
13.14	Processing new Failures	740
13.15	Class auto_ptr and Dynamic Memory Allocation	744
13.16	Standard Library Exception Hierarchy	746
14	File Processing	757
14.1	Introduction	758

14.2	The Data Hierarchy	758
14.3	Files and Streams	760
14.4	Creating a Sequential Access File	761
14.5	Reading Data from a Sequential Access File	765
14.6	Updating Sequential Access Files	771
14.7	Random-Access Files	772
14.8	Creating a Random-Access File	773
14.9	Writing Data Randomly to a Random-Access File	775
14.10	Reading Data Sequentially from a Random-Access File	777
14.11	Example: A Transaction Processing Program	779
14.12	Input/Output of Objects	785
15	Data Structures	794
15.1	Introduction	795
15.2	Self-Referential Classes	796
15.3	Dynamic Memory Allocation	797
15.4	Linked Lists	798
15.5	Stacks	811
15.6	Queues	815
15.7	Trees	818
16	Bits, Characters, Strings and Structures	849
16.1	Introduction	850
16.2	Structure Definitions	850
16.3	Initializing Structures	852
16.4	Using Structures with Functions	853
16.5	typedef	853
16.6	Example: High-Performance Card-shuffling and Dealing Simulation	854
16.7	Bitwise Operators	856
16.8	Bit Fields	865
16.9	Character-handling Library	868
16.10	String Conversion Functions	874
16.11	Search Functions of the String-handling Library	879
16.12	Memory Functions of the String-handling Library	884
16.13	Another Function of the String-handling Library	888
17	The Preprocessor	902
17.1	Introduction	903
17.2	The #include Preprocessor Directive	903
17.3	The #define Preprocessor Directive: Symbolic Constants	904
17.4	The #define Preprocessor Directive: Macros	905
17.5	Conditional Compilation	906
17.6	The #error and #pragma Preprocessor Directives	908
17.7	The # and ## Operators	908
17.8	Line Numbers	908
17.9	Predefined Symbolic Constants	909
17.10	Assertions	909

18	C Legacy Code Topics	915
18.1	Introduction	916
18.2	Redirecting Input/Output on UNIX and DOS Systems	916
18.3	Variable-Length Argument Lists	917
18.4	Using Command-Line Arguments	919
18.5	Notes on Compiling Multiple-Source-File Programs	921
18.6	Program Termination with <code>exit</code> and <code>atexit</code>	923
18.7	The <code>volatile</code> Type Qualifier	924
18.8	Suffixes for Integer and Floating-Point Constants	925
18.9	Signal Handling	925
18.10	Dynamic Memory Allocation with <code>calloc</code> and <code>realloc</code>	927
18.11	The Unconditional Branch: <code>goto</code>	928
18.12	Unions	929
18.13	Linkage Specifications	933
19	Class <code>string</code> and String Stream Processing	940
19.1	Introduction	941
19.2	<code>string</code> Assignment and Concatenation	943
19.3	Comparing <code>strings</code>	945
19.4	Substrings	947
19.5	Swapping <code>strings</code>	948
19.6	<code>string</code> Characteristics	949
19.7	Finding Characters in a <code>string</code>	951
19.8	Replacing Characters in a <code>string</code>	954
19.9	Inserting Characters into a <code>string</code>	955
19.10	Conversion to C-Style <code>char *</code> Strings	957
19.11	Iterators	959
19.12	String Stream Processing	960
20	Standard Template Library (STL)	970
20.1	Introduction to the Standard Template Library (STL)	972
20.1.1	Introduction to Containers	974
20.1.2	Introduction to Iterators	978
20.1.3	Introduction to Algorithms	983
20.2	Sequence Containers	985
20.2.1	<code>vector</code> Sequence Container	986
20.2.2	<code>list</code> Sequence Container	993
20.2.3	<code>deque</code> Sequence Container	997
20.3	Associative Containers	999
20.3.1	<code>multiset</code> Associative Container	1000
20.3.2	<code>set</code> Associative Container	1003
20.3.3	<code>multimap</code> Associative Container	1004
20.3.4	<code>map</code> Associative Container	1006
20.4	Container Adapters	1008
20.4.1	<code>stack</code> Adapter	1008
20.4.2	<code>queue</code> Adapter	1010

20.4.3	priority_queue Adapter	1012
20.5	Algorithms	1013
20.5.1	fill , fill_n , generate and generate_n	1014
20.5.2	equal , mismatch and lexicographical_compare	1016
20.5.3	remove , remove_if , remove_copy and remove_copy_if	1019
20.5.4	replace , replace_if , replace_copy and replace_copy_if	1022
20.5.5	Mathematical Algorithms	1025
20.5.6	Basic Searching and Sorting Algorithms	1028
20.5.7	swap , iter_swap and swap_ranges	1031
20.5.8	copy_backward , merge , unique and reverse	1032
20.5.9	inplace_merge , unique_copy and reverse_copy	1035
20.5.10	Set Operations	1037
20.5.11	lower_bound , upper_bound and equal_range	1040
20.5.12	Heapsort	1043
20.5.13	min and max	1046
20.5.14	Algorithms Not Covered in This Chapter	1046
20.6	Class bitset	1048
20.7	Function Objects	1052
21	Standard C++ Language Additions	1067
21.1	Introduction	1068
21.2	bool Data Type	1068
21.3	static_cast Operator	1070
21.4	const_cast Operator	1072
21.5	reinterpret_cast Operator	1073
21.6	namespaces	1074
21.7	Run-Time Type Information (RTTI)	1078
21.8	Operator Keywords	1082
21.9	explicit Constructors	1083
21.10	mutable Class Members	1089
21.11	Pointers to Class Members (.* and ->*)	1090
21.12	Multiple Inheritance and virtual Base Classes	1092
21.13	Closing Remarks	1097
A	Operator Precedence Chart	1102
B	ASCII Character Set	1104
C	Number Systems	1105
C.1	Introduction	1106
C.2	Abbreviating Binary Numbers as Octal Numbers and Hexadecimal Numbers	1109
C.3	Converting Octal Numbers and Hexadecimal Numbers to Binary Numbers	1110
C.4	Converting from Binary, Octal, or Hexadecimal to Decimal	1110
C.5	Converting from Decimal to Binary, Octal, or Hexadecimal	1111
C.6	Negative Binary Numbers: Two's Complement Notation	1112

D	C++ Internet and Web Resources	1118
D.1	Resources	1118
D.2	Tutorials	1119
D.3	FAQs	1119
D.4	Visual C++	1120
D.5	comp.lang.c++	1120
D.6	Compilers	1122
D.7	Development Tools	1122
D.8	Standard Template Library	1123
	Bibliography	1125
	Index	1131