



Table of Contents

1	Writing an ANSI C++ Program	1
1.1	Getting Ready to Program	2
1.1	A First Program	3
1.2	Problem Solving: Recipes	7
1.2.1	Algorithms—Being Precise.	8
1.3	Implementing Our Algorithm in C++	10
1.4	Software Engineering: Style	12
1.5	Common Programming Errors	13
1.6	Writing and Running a C++ Program	14
1.6.1	Interrupting a Program	16
1.6.2	Typing an End-of-File Signal	16
1.7	Dr. P's Prescriptions	16
1.8	C++ Compared with Java	17
	Summary	20
	Review Questions	21
	Exercises	22

2	Native Types and Statements	25
2.1	Program Elements	26
2.1.1	Comments	26
2.1.2	Keywords	27
2.1.3	Identifiers	27
2.1.4	Literals	29
2.1.5	Operators and Punctuators	31
2.2	Input/Output	31
2.3	Program Structure	34
2.3.1	Redirection	36
2.4	Simple Types	37
2.4.1	Initialization	39
2.5	The Traditional Conversions	40
2.6	Enumeration Types	43
2.6.1	<code>typedef</code> Declarations	44
2.7	Expressions	44
2.7.1	Precedence and Associativity of Operators	45
2.7.2	Relational, Equality, and Logical Operators	47
2.8	Statements	50
2.8.1	Assignment and Expressions	51
2.8.2	The Compound Statement	52
2.8.3	The <code>if</code> and <code>if-else</code> Statements	52
2.8.4	The <code>while</code> Statement	55
2.8.5	The <code>for</code> Statement	56
2.8.6	The <code>do</code> Statement	57
2.8.7	The <code>break</code> and <code>continue</code> Statements	58
2.8.8	The <code>switch</code> Statement	59
2.8.9	The <code>goto</code> Statement	62
2.9	Software Engineering: Debugging	62
2.10	Dr. P's Prescriptions	65
2.11	C++ Compared with Java	67
	Summary	69
	Review Questions	70
	Exercises	71

3	Functions, Pointers, and Arrays	75
3.1	Functions	75
3.2	Function Invocation	76
3.3	Function Definition	78
3.4	The <code>return</code> Statement	79
3.5	Function Prototypes	80
3.6	Call-By-Value	81
3.7	Recursion	83
3.8	Default Arguments	84
3.9	Functions as Arguments	86
3.10	Overloading Functions	88
3.11	Inlining	89
3.11.1	Software Engineering: Avoiding Macros	89

3.12	Scope and Storage Class	90
3.12.1	The Storage Class <code>auto</code>	92
3.12.2	The Storage Class <code>extern</code>	92
3.12.3	The Storage Class <code>register</code>	93
3.12.4	The Storage Class <code>static</code>	94
3.12.5	Header Files and Linkage Mysteries.	95
3.13	Namespaces	98
3.14	Pointer Types	99
3.14.1	Addressing and Dereferencing	100
3.14.2	Pointer-Based Call-By-Reference.	100
3.15	Reference Declarations	102
3.16	The Uses of <code>void</code>	104
3.17	Arrays	105
3.17.1	Subscripting	106
3.17.2	Initialization	106
3.18	Arrays and Pointers	106
3.19	Passing Arrays to Functions.	107
3.20	Problem Solving: Random Numbers.	108
3.21	Software Engineering: Structured Programming.	111
3.22	Core Language ADT: <code>char*</code> String.	114
3.23	Multidimensional Arrays	117
3.24	Operators <code>new</code> and <code>delete</code>	120
3.24.1	Vector Instead of Array	123
3.24.2	String Instead of <code>char*</code>	124
3.25	Software Engineering: Program Correctness.	124
3.26	Dr. P's Prescriptions	127
3.27	C++ Compared with Java	128
	Summary.	130
	Review Questions	132
	Exercises.	133

4 Classes and Abstract Data Types 139

4.1	The Aggregate Type <code>class</code> and <code>struct</code>	140
4.2	Member Selection Operator	141
4.3	Member Functions.	143
4.4	Access: Private and Public	146
4.5	Classes	147
4.6	Class Scope.	150
4.6.1	Scope Resolution Operator	150
4.6.2	Nested Classes	152
4.7	An Example: Flushing	153
4.8	The <code>this</code> Pointer	158
4.9	<code>static</code> Members.	159
4.10	<code>const</code> Members.	161
4.10.1	Mutable Members	163
4.11	A Container Class Example: <code>ch_stack</code>	164

4.12	Software Engineering: Class Design	166
4.12.1	Trade-Offs in Design	168
4.12.2	Unified Modeling Language (UML) and Design	169
4.13	Dr. P's Prescriptions	170
4.14	C++ Compared with Java	171
4.15	Advanced Topics	172
4.15.1	Pointer to Class Member	172
4.15.2	Unions	174
4.15.3	Bit Fields	175
	Summary	177
	Review Questions	178
	Exercises	179

5 Ctors, Dtors, Conversions, and Operator Overloading 183

5.1	Classes with Constructors	184
5.1.1	The Default Constructor	186
5.1.2	Constructor Initializer	187
5.1.3	Constructors as Conversions	187
5.1.4	Improving the <code>point</code> Class	189
5.1.5	Constructing a Stack	190
5.1.6	The Copy Constructor	193
5.2	Classes with Destructors	195
5.3	Members That Are Class Types	195
5.4	Example: A Singly Linked List	196
5.5	Strings Using Reference Semantics	201
5.6	Constructor Issues and Mysteries	204
5.6.1	Destructor Details	205
5.6.2	Constructor Pragmatics	206
5.7	Polymorphism Using Function Overloading	206
5.8	ADT Conversions	207
5.9	Overloading and Signature Matching	208
5.10	Friend Functions	211
5.11	Overloading Operators	213
5.12	Unary Operator Overloading	214
5.13	Binary Operator Overloading	217
5.14	Overloading the Assignment Operator	219
5.15	Overloading the Subscript Operator	220
5.16	Overloading Operator <code>()</code> for Indexing	221
5.17	Overloading <code><<</code> and <code>>></code>	222
5.18	Overloading <code>-></code>	223
5.19	Overloading <code>new</code> and <code>delete</code>	224
5.20	More Signature Matching	227
5.21	Software Engineering: When to Use Overloading	228
5.22	Dr. P's Prescriptions	229
5.23	C++ Compared with Java	231
	Summary	235
	Review Questions	236
	Exercises	237

6	Templates and Generic Programming	243
6.1	Template Class <code>stack</code>	246
6.2	Function Templates	248
6.2.1	Signature Matching and Overloading	250
6.2.2	How to Write a Simple Function: <code>square()</code>	252
6.3	Generic Code Development: Quicksort	253
6.3.1	Converting to a Generic <code>quicksort()</code>	256
6.4	Class Templates	260
6.4.1	Friends	260
6.4.2	Static Members	260
6.4.3	Class Template Arguments	261
6.4.4	Default Template Arguments	261
6.4.5	Member Templates	262
6.5	Parameterizing the Class <code>vector</code>	262
6.6	Using STL: <code>string</code> , <code>vector</code> , and <code>complex</code>	265
6.6.1	<code>string</code> and <code>basic_string<></code>	265
6.6.2	<code>vector<></code> in STL	267
6.6.3	Using <code>complex<></code>	267
6.6.4	<code>limits</code> and Other Useful Templates	268
6.7	Software Engineering: Reuse and Generics	269
6.7.1	Debugging Template Code	269
6.7.2	Special Considerations	270
6.7.3	Using <code>typename</code>	271
6.8	Dr. P's Prescriptions	272
6.9	C++ Compared with Java	272
	Summary	276
	Review Questions	277
	Exercises	278

7	Standard Template Library	280
7.1	A Simple STL Example	280
7.2	Containers	283
7.2.1	Sequence Containers	285
7.2.2	Associative Containers	288
7.2.3	Container Adaptors	293
7.3	Iterators	296
7.3.1	Iterators for <code>istream</code> and <code>ostream</code>	297
7.3.2	Iterator Adaptors	300
7.4	Algorithms	302
7.4.1	Sorting Algorithms	302
7.4.2	Nonmutating Sequence Algorithms	305
7.4.3	Mutating Sequence Algorithms	307
7.4.4	Numerical Algorithms	310
7.5	Numerical Integration Made Easy	311
7.6	STL: Function Objects	315
7.6.1	Building a Function Object	317
7.6.2	Function Adaptors	318
7.7	Allocators	320

7.8	Software Engineering: STL Use	320
7.8.1	Syntax Bugs	321
7.9	Dr. P's Prescriptions	322
7.10	C++ Compared with Java	323
	Summary	324
	Review Questions	325
	Exercises	326

8 Inheritance and OOP 328

8.1	A Derived Class	329
8.1.1	More Unified Modeling Language (UML).	333
8.2	A Student ISA Person	334
8.3	Virtual Functions: Dynamic Determination	337
8.3.1	Overloading and Overriding Confusion	340
8.3.2	A Canonical Example: Class <code>shape</code>	342
8.4	Abstract Base Classes	343
8.5	Templates and Inheritance.	350
8.6	Multiple Inheritance.	351
8.7	RTTI and Other Fine Points	353
8.7.1	Finer Points.	354
8.8	Software Engineering: Inheritance and Design	355
8.8.1	Subtyping Form.	356
8.8.2	Code Reuse.	357
8.9	Dr. P's Prescriptions	358
8.10	C++ Compared with Java	358
	Summary	361
	Review Questions	362
	Exercises	363

9 Input/Output 366

9.1	The Output Class <code>ostream</code>	366
9.2	Formatted Output and <code>omanip</code>	367
9.3	User-Defined Types: Output.	372
9.4	The Input Class <code>istream</code>	374
9.5	Files	375
9.6	Using Strings as Streams	379
9.7	The Functions and Macros in <code>ctype</code>	380
9.8	Using Stream States.	380
9.9	Mixing I/O Libraries.	383
9.10	Software Engineering: I/O	384
9.11	Dr. P's Prescriptions	386
9.12	C++ Compared with Java	386
	Summary	389
	Review Questions	390
	Exercises	391

10	Exceptions and Program Correctness	394
10.1	Using the <i>assert</i> Library	394
10.2	C++ Exceptions	397
10.3	Throwing Exceptions	398
	10.3.1 Rethrown Exceptions	400
	10.3.2 Exception Expressions	401
10.4	<i>try</i> Blocks	404
10.5	Handlers	405
10.6	Converting Assertions to Exceptions	405
10.7	Exception Specification	408
10.8	<i>terminate()</i> and <i>unexpected()</i>	409
10.9	Standard Exceptions and Their Uses	409
10.10	Software Engineering: Exception Objects	411
10.11	Dr. P's Prescriptions	413
10.12	C++ Compared with Java	414
	Summary	417
	Review Questions	418
	Exercises	419

11	OOP Using C++	421
11.1	OOP Language Requirements	422
	11.1.1 ADTs: Encapsulation and Data Hiding	423
	11.1.2 Reuse and Inheritance	423
	11.1.3 Polymorphism	424
11.2	OOP: The Dominant Programming Methodology	425
11.3	Designing with OOP in Mind	432
11.4	Class-Responsibility-Collaborator	434
	11.4.1 CRC Cards	435
11.5	Design Patterns	436
11.6	A Further Assessment of C++	437
	11.6.1 Why C++ Is Better Than Java	438
	11.6.2 A Short Rebuttal	439
11.7	Software Engineering: Last Thoughts	439
11.8	Dr. P's Prescriptions	440
11.9	C++ Compared with Java	441
	Summary	447
	Review Questions	448
	Exercises	449

A	ASCII Character Codes	451
B	Operator Precedence and Associativity	453
C	String Library	455
C.1	Constructors	456
C.2	Member Functions	457
C.3	Global Operators	460
D	The tio Library	462
D.1	Console	462
D.2	FormattedWriter	463
D.3	PrintFileWriter	472
D.4	ReadException	472
D.5	ReadInput	473
	Index	482