

Contents

Preface	xiii
Acknowledgements	xv
Introduction	xvii
Studying C++	xviii
Using This Book	xix
A Comment on Comments	xx
Overview of C++	1
What is in a Name	1
What is in C++	1
Different Backgrounds	3
Fundamental C++ for C++ Programmers	3
Fundamental C++ for C Programmers	4
Fundamental C++ for Java Programmers	5
Fundamental C++ for C# Programmers	5
Fundamental C++ for COBOL Programmers	6
Fundamental C++ for Python Programmers	6
Fundamental C++ for (Visual) Basic Programmers	7
Fundamental C++ for Pascal and Delphi Programmers	7
Fundamental C++ for Functional Programmers	8
Fundamental C++ for Lisp and Logo Programmers	8
Fundamental C++ for Object-Oriented Programmers	9
Fundamental C++ for Every Programmer	9
1 Getting Started	11
Creating a ‘Hello World’ Program	12
What the Code Means	16
Our Second Program—An Empty Playpen	17
What the Code Means	20

Something to Play With	21
Summary	21
2 Fundamental Types, Operators, and Simple Variables	23
A Simple Program	23
What Is a Type?	24
What Are Fundamental Types?	25
Representing Negative Integers	26
Derivative Types	27
Declaration and Definition	28
Names in C++	28
Operators	29
A Simple Program	30
Exceptions – Handling Bad Input	31
Writing Correct Code	32
Getting Output Before Handling an Exception	33
A Little More About Playpen	34
Default Playpen Color Names	36
Characters and Text	37
Floating-Point Numbers	39
First Floating-Point Program	39
3 Looping and Making Decisions	51
Some Library Types	51
Making Decisions	53
Looping	58
On Magic Numbers	65
4 Namespaces and the C++ Standard Library	71
Wide Versus Narrow Character Set Support	71
Namespaces	72
Input from <code>std::cin</code>	76
Output with <code>std::cout</code>	78
Standard Console Output Objects	78
Playpen Plotting Modes	79
Further Practice	80
5 Writing Functions in C++	85
The C++ Function Concept	85
Sorting in Other Orders	86
Designing a Function	87
C++ Procedures	92
Pure Functions	92
Overloading Functions	93
Resetting <code>istream</code> and <code>ostream</code> Objects	96
Unnamed Parameters	103
Separate Compilation and Header Files	104

6 Behavior, Sequence Points, and Order of Evaluation	109
Types of Behavior	109
Sequence Points	113
Order of Evaluation	115
Guidelines	116
 7 Generic Functions	 119
Which Is Larger	119
Getting the Largest	121
Getting the Largest Using a typedef	121
Getting the Largest Using a Template	123
Ambiguity	126
Function Templates Can Be Specialized	128
Specializing max()	129
Overloading Function Templates	130
C++ Iterators	132
Version of max(std::vector) Using an Iterator	133
The fgw::read Function Templates	134
 8 User-Defined Types, Part 1: typedef and enum	 143
typedef : New Names for Old	143
On Reading Declarations	145
enum	147
Operator Overloading	150
 9 User-Defined Types, Part 2: Simple classes (value types)	 157
ISBN as a class Type	158
Testing Code	162
Overloading Operators	164
A Value Type for Playing Cards	165
public Versus private	166
Special Member Functions: Constructors	166
Special Member Functions: Destructors	167
Special Member Functions: Copy assignment, operator=	167
Ordinary Member Functions	168
Implementing Constructors	168
Implementing a Destructor	169
Implementing Copy Assignment, operator=	169
Implementing a Member Function	170
Separate Compilation	171
Developing the card_value Type	174
Changing the Implementation	177
Pointers and Arrays	179
Consolidation – a Point Class	181
Defining Member Functions in a Class Definition	184
Constructors and Destructors	187
 10 User-Defined Types, Part 3: Simple classes (homogeneous entity types)	 189
Examples of Value and Entity Types	189

A Simple Playing-Card Entity	190
Another Entity Type: Deck of Cards	192
Output for deck	195
Creating a deck Instance From a File	198
11 Pointers, Smart Pointers, Iterators, and Dynamic Instances	203
Raw Pointers	203
A Dangerous Special Case	205
Arrays	206
Arrays and Pointers	208
Dynamic Instances	209
Smart Pointers	216
Iterators	217
12 User-Defined Types, Part 4: Class hierarchies, polymorphism, inheritance, and subtypes	221
An Interface for a Chess Piece	222
Implementing basic_chesspiece	224
Implementing a Knight	228
Getting Polymorphic Behavior	231
Getting the Identity	232
Removing an Irritant	233
Moving to an Occupied Square	234
Another Piece	234
13 Dynamic Object Creation and Polymorphic Objects	239
Selecting the Subtype at Runtime	239
Unnamed Namespaces	241
A Chess-Piece Type	244
Implementing chesspiece	246
Defining and Implementing the Subtypes	248
Constructing a Specific Chess Piece	251
The chesspiece Constructor and transform()	252
Implementing the Rest of chesspiece	252
Collections of Objects	255
Design and Implementation of a chessboard Type	256
14 Streams, Files, and Persistence	259
The C++ Stream Hierarchy	259
Appending Data	262
Consolidation	263
String Streams	263
Converting Numerical Values to Strings	265
Persistence	266
Converting Text to an Enumerator	268
15 Exceptions	273
What Is an Exception?	273
What Can I throw ?	275

The Exception-Safe Copy-Assignment Idiom	281
Rethrowing	282
Exception Specifications: An Idea That Failed	283
Exceptions and Destructors	283
 16 Overloading Operators and Conversion Operators	 285
Overloading Operators for an Arithmetic Type	285
Conversion Operators	289
Function Objects	291
Conclusion	294
 17 Containers, Iterators, and Algorithms	 297
Working with a Set	298
Working with Numeric Algorithms	303
Working with a Multimap	306
Preloading a Container	307
Conclusion	308
 18 Something Old, Something New	 313
Code Layout and Consistency	313
Where to Put <code>const</code>	314
Function-Style Versus Assignment-Style Initialization	315
Using <code>using</code>	318
Switching Off Polymorphism	319
Alternative Spellings for Operators	320
Hungarian Notation	320
Names for Constants	320
Comments and ‘Need to Know’	321
Multiple Exits from Structures	321
Refactoring and the Power of Objects	323
Using a Legacy Library	327
In Conclusion	329
 Appendix A: Those Who Went Before	 331
References	349
Index	351