Table of Contents

Preface **1**

1 Modelling Functions **5**
1.1 Goals and Objectives 5
1.2 What is a Function and why do we need Functions? 6
1.3 Function Pointers in C 7
1.4 Function Essentials 9
1.5 Modelling Algebraic Expressions using Function 11
1.6 Generic Functions and Functions as Data Members 13
1.7 Callback Functions and Notification Patterns 15
1.8 Class-based Callback Functions 17
1.9 Scalar, Vector and Vector-valued Functions 18
1.9.1 Scalar-valued Functions 20
1.9.2 Vector Functions 21
1.9.3 Vector-valued Functions 21
1.10 Conclusions and Summary 22

2 Bind **23**
2.1 Introduction and Objectives 23
2.2 An Introduction to Function Objects 23
2.3 Predefined and User-defined Function Objects 26
2.4 The `for_each()` Algorithm 28
2.5 Function Adapters for Member Functions and Ordinary Functions 29
2.6 Introducing Bind 31
2.7 Placeholders and Arguments 31
2.8 Calling a Member Function 32
2.9 Using Bind with Function Objects 33
2.10 Dynamic Sorting and Bind 34
2.11 Function Composition 35
2.12 Applications and Relationships with STL and Boost 36
2.13 Summary and Conclusions 38

3 Event Notification, Observer Pattern and Signals Library **39**
3.1 Introduction and Objectives 39
3.2 Notification Patterns in Object-Oriented Systems 39
3.3 The GOF Observer Pattern: what is it and what are the Problems? 40
3.3.1 The GOF Observer Pattern: Critique 44
3.4 An Introduction to the Signals Library 44
3.5 Signals and Slots 45
3.6 Slot Groups 46
3.7 Objects playing Roles of Signals and Slots 47
3.8 Connection Management and the `connection` Class 49
3.9 Creating Slots using Bind and Lambda 51
3.9.1 Creating Slots 'on the fly' with Lambda 51
3.9.2 Creating Slot Types with Bind 52
3.10 Combiners 53
3.11 Thread-Safe Signals Library 54
3.12 The GOF Observer Pattern revisited 55
3.13 Summary and Conclusions 56

4 Phoenix **57**
4.1 Introduction and Objectives 57

4.2 Motivating Phoenix: a Test Case 57
4.3 Introduction to Functional Programming 59
4.3.1 Lambda Calculus and Lambda Functions 60
4.4 The Architecture of Phoenix 60
4.5 Actors, Arguments and Fundamentals 61
4.6 Composite 63
4.7 Function and Operator 63
4.8 Statement 65
4.9 Object 67
4.10 Scope and Local Variables 67
4.11 Bind 68
4.12 Container 70
4.13 Algorithm 70
4.14 Summary and Conclusions 71

5 Smart Pointers and Serialisation 72
5.1 Introduction and Objectives 72
5.2 An Introduction to Memory Management 72
5.3 An Introduction to Smart Pointers 73
5.4 Scoped Pointers and Scoped Arrays 74
5.5 Shared Pointer and Shared Arrays 76
5.5.1 Functionality of `shared_ptr` 77
5.5.2 Stop Sharing Ownership 78
5.5.3 Retrieving the Resource from Shared Data 78
5.5.4 Shared Arrays 79
5.5.5 Custom Deleters 79
5.6 Weak Pointers 81
5.7 Smart Pointers and Exception Handling 82
5.8 Using Smart Pointers with STL Containers 82
5.9 Test Case: creating smart Composite Objects 84
5.10 Serialisation and Object Persistence 85
5.10.1 Main Concepts and Initial Examples 86
5.10.2 STL Containers and Serialisation of more Complex Objects 88
5.10.3 XML Serialisation and Deserialisation 89
5.10.4 An Example from Boost Documentation: Serialisation Tutorial 91
5.11 Summary and Conclusions 95

6 Tuple 97
6.1 Introduction and Objectives 97
6.2 An Introduction to n-Tuples 97
6.3 The `tuple` Class: Fundamental Properties 98
6.4 Accessing Tuple Elements 99
6.5 Comparing Tuples 100
6.6 Tuples and Streaming 101
6.7 Applications of Tuple 102
6.8 Summary and Conclusions 103

7 Any 104
7.1 Introduction and Objectives 104
7.2 The `any` Class: Fundamental Properties 104
7.3 Using `any` with Smart Pointers 106
7.4 STL Containers whose Elements are of Type `any` 106

7.5 Property Sets with heterogeneous Elements 107
7.6 Initialisation of Data with Assign Library 111
7.7 Summary and Conclusions 114

8 Variant 115
8.1 Introduction and Objectives 115
8.2 What is a Discriminated Union? 115
8.3 The `variant` Class Template: Hello World Example....116
8.4 Member Functions in `variant`....116
8.5 Type-safe Visitation of Variants 117
8.6 Summary and Conclusions 118

9 Number Systems....120
9.1 Introduction and Objectives 120
9.2 A Review of STL `complex<T>` Template Class 120
9.2.1 Application: Discrete Fourier Transform (DFT) 121
9.3 Complex Numbers Algorithms 124
9.4 Rational Numbers....125
9.5 Greatest Common Divisor (gcd) and Least Common Multiple (lcm)....126
9.5.1 More general Cases 128
9.6 Quaternions and Octonians....129
9.7 Conversions 131
9.8 Summary and Conclusions 133

10 String Algorithm 134
10.1 Introduction and Objectives 134
10.2 Case Conversion, Trimming and Predicates 134
10.3 Find Algorithms 136
10.4 Erase and Replace 138
10.5 Split and Join 138
10.6 Finders and Formatters 139
10.7 Iterators....142
10.7.1 Creating Custom Formatters 143
10.8 Classification 144
10.9 Creating Name-Value Maps 145
10.10 The Range Library 145
10.11 Summary and Conclusions 147

11 Tokenizer....148
11.1 Introduction and Objectives 148
11.2 What is a Token?....148
11.3 The Token Class, Token Iterator and Token Function 149
11.3.1 Token Class....149
11.3.2 TokenizerFunction Concept....150
11.3.3 Token Iterator....150
11.3.4 Examples 150
11.4 Conversions and Casting Examples 151
11.5 Summary and Conclusions 152

12 Regex....154
12.1 Introduction and Objectives 154
12.2 Alphabet, Words and Language....154
12.3 An Introduction to Regular Expressions....155
12.4 Regex Functionality 156

12.5 The Class `basic_regex` 156
12.6 Regular Expression Matching 158
12.7 Searching 159
12.8 Text Replacement 161
12.9 Dynamic Regular Expressions and Exception Handling 161
12.10 Regex and Callbacks 162
12.11 Summary and Conclusions 163

13 An Introduction to Expression Templates and Xpressive 164
13.1 Introduction and Objectives 164
13.2 Binary Trees and Expression Trees 164
13.2.1 Traversing Binary Trees 166
13.2.2 Extended Binary Trees 167
13.3 An Introduction to Expression Templates 169
13.4 An Introduction to Xpressive 173
13.5 First Encounters with Xpressive 173
13.6 Regex Object and Matches 174
13.7 Nested Regex and simple Grammar 176
13.8 Semantic Actions 178
13.9 More Semantic Actions 180
13.10 Conclusions and Summary 181

14 MultiArray and Array 182
14.1 Introduction and Objectives 182
14.2 A Quick '101' Tour of MultiArray 183
14.3 Overview of MultiArray Functionality 184
14.4 Specifying Array Dimensions 184
14.5 Accessing the Elements of a Multi-Array 185
14.6 Setting the Array Base 186
14.7 Storage Ordering 187
14.8 Array 188
14.9 Applications of Boost.Array 190
14.10 Four-dimensional Arrays 192
14.11 Views and Slices 192
14.11.1 Zooming 192
14.11.2 Slicing 194
14.12 MultiArray Adapters 197
14.13 Utility Print Functions 197
14.14 MultiArrays in Combination with other Data Structures 198
14.15 Summary and Conclusions 200

15 Random Number Generation 201
15.1 Introduction and Objectives 201
15.2 Overview of Random Number Generation 201
15.3 Random Number Generators in Boost 202
15.4 variate_generator 205
15.5 Performance, Reliability, Suitability and Accuracy Requirements 206
15.6 Examples using Random Number Generators 207
15.6.1 Calculating π 207
15.6.2 Finding Real Roots of a Quadratic Equation 208
15.6.3 Kernel Density Estimation 209
15.7 Conclusions and Summary 211

16 Flyweight and Functional/Hash 212
16.1 Introduction and Objectives 212
16.2 The "Hello World" Example 212
16.3 Flyweight Requirements 214
16.4 Key-Value Flyweights 217
16.5 Flyweight Factory Specification 219
16.6 Tracking and Lifecycle Policies 220
16.7 Tagging Policies 221
16.8 Holders 222
16.9 Locking Policies and Thread-Safe Code 223
16.10 Summary and Conclusions 224
16.11 Appendix: Functional/Hash 224

17 Integrating Legacy Applications with Boost 229
17.1 Introduction and Objectives 229
17.2 Migrating Legacy Code to Boost: The CADObject Project 229
17.3 CADObject Architecture: Version One 230
17.4 CADObject Architecture: Version Two 234
17.5 CADObject Architecture: Version Three 236
17.6 Using Boost.Serialization 237
17.7 Factories and Deserialisation 240
17.8 Conclusions and Summary 242

18 An Introduction to Thread 243
18.1 Introduction and Objectives 243
18.2 An Introduction to Threads 243
18.3 The Life of a Thread 244
18.3.1 How Threads Communicate 244
18.4 What Kinds of Applications are suitable for Multi-Threading? 244
18.4.1 Suitable Tasks for Multi-threading 245
18.5 The Boost `thread` class 245
18.6 The Life of a Thread 248
18.7 Basic Thread Synchronisation 251
18.8 Thread Interruption 253
18.9 Thread Notification 254
18.10 Thread Groups 254
18.11 Shared Queue Pattern 255
18.12 The Producer-Consumer Pattern 256
18.13 Thread Local Storage 258
18.14 Summary and Conclusions 258

Appendix A: Generic Programming 260
Introduction and Objectives 260
Some Useful Techniques 260
Traits Classes 262
An Introduction to Policy-based Design 264
Curiously Recurring Template Pattern (CRTP) 267
The Boost Categories 270

Appendix B: Boost, Design Patterns and Applications 273
Introduction and Objectives 273
An Overview of Design Patterns 273
Boost and Design Patterns 274

Which Boost Libraries 'Bootstrap' the GOF Design Patterns? 275
Creating Layered Software Systems 276

Appendix C: Exercises and Projects 278
Introduction and Objectives 278

Epilogue and Volume II Contents 290

Bibliography 291

Index 292

Book Registration Form 297