

Contents

Preface	xv
Acknowledgments	xvii
1 About This Book	1
1.1 What You Should Know Before Reading This Book	2
1.2 Overall Structure of the Book	2
1.3 How to Read This Book	3
1.4 Some Remarks About Programming Style	3
1.5 The Standard versus Reality	5
1.6 Example Code and Additional Informations	5
1.7 Feedback	5
Part I: The Basics	7
2 Function Templates	9
2.1 A First Look at Function Templates	9
2.1.1 Defining the Template	9
2.1.2 Using the Template	10
2.2 Argument Deduction	12
2.3 Template Parameters	13
2.4 Overloading Function Templates	15
2.5 Summary	19
3 Class Templates	21
3.1 Implementation of Class Template Stack	21
3.1.1 Declaration of Class Templates	22

3.1.2	Implementation of Member Functions	24
3.2	Use of Class Template <code>Stack</code>	25
3.3	Specializations of Class Templates	27
3.4	Partial Specialization	29
3.5	Default Template Arguments	30
3.6	Summary	33
4	Nontype Template Parameters	35
4.1	Nontype Class Template Parameters	35
4.2	Nontype Function Template Parameters	39
4.3	Restrictions for Nontype Template Parameters	40
4.4	Summary	41
5	Tricky Basics	43
5.1	Keyword <code>typename</code>	43
5.2	Using <code>this-></code>	45
5.3	Member Templates	45
5.4	Template Template Parameters	50
5.5	Zero Initialization	56
5.6	Using String Literals as Arguments for Function Templates	57
5.7	Summary	60
6	Using Templates in Practice	61
6.1	The Inclusion Model	61
6.1.1	Linker Errors	61
6.1.2	Templates in Header Files	63
6.2	Explicit Instantiation	65
6.2.1	Example of Explicit Instantiation	65
6.2.2	Combining the Inclusion Model and Explicit Instantiation	67
6.3	The Separation Model	68
6.3.1	The Keyword <code>export</code>	68
6.3.2	Limitations of the Separation Model	70
6.3.3	Preparing for the Separation Model	70
6.4	Templates and <code>inline</code>	72
6.5	Precompiled Headers	72
6.6	Debugging Templates	74

Contents	vii
6.6.1 Decoding the Error Novel	75
6.6.2 Shallow Instantiation	77
6.6.3 Long Symbols	79
6.6.4 Tracers	79
6.6.5 Oracles	84
6.6.6 Archetypes	85
6.7 Afternotes	85
6.8 Summary	85
7 Basic Template Terminology	87
7.1 “Class Template” or “Template Class”?	87
7.2 Instantiation and Specialization	88
7.3 Declarations versus Definitions	89
7.4 The One-Definition Rule	90
7.5 Template Arguments versus Template Parameters	90
Part II: Templates in Depth	93
8 Fundamentals in Depth	95
8.1 Parameterized Declarations	95
8.1.1 Virtual Member Functions	98
8.1.2 Linkage of Templates	99
8.1.3 Primary Templates	100
8.2 Template Parameters	100
8.2.1 Type Parameters	101
8.2.2 Nontype Parameters	101
8.2.3 Template Template Parameters	102
8.2.4 Default Template Arguments	103
8.3 Template Arguments	104
8.3.1 Function Template Arguments	105
8.3.2 Type Arguments	108
8.3.3 Nontype Arguments	109
8.3.4 Template Template Arguments	111
8.3.5 Equivalence	113
8.4 Friends	113
8.4.1 Friend Functions	114

8.4.2	Friend Templates	117
8.5	Afternotes	117
9	Names in Templates	119
9.1	Name Taxonomy	119
9.2	Looking Up Names	121
9.2.1	Argument-Dependent Lookup	123
9.2.2	Friend Name Injection	125
9.2.3	Injected Class Names	126
9.3	Parsing Templates	127
9.3.1	Context Sensitivity in Nontemplates	127
9.3.2	Dependent Names of Types	130
9.3.3	Dependent Names of Templates	132
9.3.4	Dependent Names in Using-Declarations	133
9.3.5	ADL and Explicit Template Arguments	135
9.4	Derivation and Class Templates	135
9.4.1	Nondependent Base Classes	135
9.4.2	Dependent Base Classes	136
9.5	Afternotes	139
10	Instantiation	141
10.1	On-Demand Instantiation	141
10.2	Lazy Instantiation	143
10.3	The C++ Instantiation Model	146
10.3.1	Two-Phase Lookup	146
10.3.2	Points of Instantiation	146
10.3.3	The Inclusion and Separation Models	149
10.3.4	Looking Across Translation Units	150
10.3.5	Examples	151
10.4	Implementation Schemes	153
10.4.1	Greedy Instantiation	155
10.4.2	Queried Instantiation	156
10.4.3	Iterated Instantiation	157
10.5	Explicit Instantiation	159
10.6	Afternotes	163

11	Template Argument Deduction	167
11.1	The Deduction Process	167
11.2	Deduced Contexts	169
11.3	Special Deduction Situations	171
11.4	Allowable Argument Conversions	172
11.5	Class Template Parameters	173
11.6	Default Call Arguments	173
11.7	The Barton-Nackman Trick	174
11.8	Afternotes	177
12	Specialization and Overloading	179
12.1	When “Generic Code” Doesn’t Quite Cut It	179
12.1.1	Transparent Customization	180
12.1.2	Semantic Transparency	181
12.2	Overloading Function Templates	183
12.2.1	Signatures	184
12.2.2	Partial Ordering of Overloaded Function Templates	186
12.2.3	Formal Ordering Rules	188
12.2.4	Templates and Nontemplates	189
12.3	Explicit Specialization	190
12.3.1	Full Class Template Specialization	190
12.3.2	Full Function Template Specialization	194
12.3.3	Full Member Specialization	197
12.4	Partial Class Template Specialization	200
12.5	Afternotes	203
13	Future Directions	205
13.1	The Angle Bracket Hack	205
13.2	Relaxed typename Rules	206
13.3	Default Function Template Arguments	207
13.4	String Literal and Floating-Point Template Arguments	209
13.5	Relaxed Matching of Template Template Parameters	211
13.6	Typedef Templates	212
13.7	Partial Specialization of Function Templates	213
13.8	The typeof Operator	215

13.9	Named Template Arguments	216
13.10	Static Properties	218
13.11	Custom Instantiation Diagnostics	218
13.12	Overloaded Class Templates	221
13.13	List Parameters	222
13.14	Layout Control	224
13.15	Initializer Deduction	225
13.16	Function Expressions	226
13.17	Afternotes	228

Part III: Templates and Design **229**

14 The Polymorphic Power of Templates **231**

14.1	Dynamic Polymorphism	231
14.2	Static Polymorphism	234
14.3	Dynamic versus Static Polymorphism	238
14.4	New Forms of Design Patterns	239
14.5	Generic Programming	240
14.6	Afternotes	243

15 Traits and Policy Classes **245**

15.1	An Example: Accumulating a Sequence	245
15.1.1	Fixed Traits	246
15.1.2	Value Traits	250
15.1.3	Parameterized Traits	254
15.1.4	Policies and Policy Classes	255
15.1.5	Traits and Policies: What's the Difference?	258
15.1.6	Member Templates versus Template Template Parameters	259
15.1.7	Combining Multiple Policies and/or Traits	261
15.1.8	Accumulation with General Iterators	262
15.2	Type Functions	263
15.2.1	Determining Element Types	264
15.2.2	Determining Class Types	266
15.2.3	References and Qualifiers	268
15.2.4	Promotion Traits	271

Contents	xi
15.3 Policy Traits	275
15.3.1 Read-only Parameter Types	276
15.3.2 Copying, Swapping, and Moving	279
15.4 Afternotes	284
16 Templates and Inheritance	285
16.1 Named Template Arguments	285
16.2 The Empty Base Class Optimization (EBCO)	289
16.2.1 Layout Principles	290
16.2.2 Members as Base Classes	293
16.3 The Curiously Recurring Template Pattern (CRTP)	295
16.4 Parameterized Virtuality	298
16.5 Afternotes	299
17 Metaprograms	301
17.1 A First Example of a Metaprogram	301
17.2 Enumeration Values versus Static Constants	303
17.3 A Second Example: Computing the Square Root	305
17.4 Using Induction Variables	309
17.5 Computational Completeness	312
17.6 Recursive Instantiation versus Recursive Template Arguments	313
17.7 Using Metaprograms to Unroll Loops	314
17.8 Afternotes	318
18 Expression Templates	321
18.1 Temporaries and Split Loops	322
18.2 Encoding Expressions in Template Arguments	328
18.2.1 Operands of the Expression Templates	328
18.2.2 The Array Type	332
18.2.3 The Operators	334
18.2.4 Review	336
18.2.5 Expression Templates Assignments	338
18.3 Performance and Limitations of Expression Templates	340
18.4 Afternotes	341

Part IV: Advanced Applications	345
19 Type Classification	347
19.1 Determining Fundamental Types	347
19.2 Determining Compound Types	350
19.3 Identifying Function Types	352
19.4 Enumeration Classification with Overload Resolution	356
19.5 Determining Class Types	359
19.6 Putting It All Together	359
19.7 Afternotes	363
20 Smart Pointers	365
20.1 Holders and Trules	365
20.1.1 Protecting Against Exceptions	366
20.1.2 Holders	368
20.1.3 Holders as Members	370
20.1.4 Resource Acquisition Is Initialization	373
20.1.5 Holder Limitations	373
20.1.6 Copying Holders	375
20.1.7 Copying Holders Across Function Calls	375
20.1.8 Trules	376
20.2 Reference Counting	379
20.2.1 Where Is the Counter?	380
20.2.2 Concurrent Counter Access	381
20.2.3 Destruction and Deallocation	382
20.2.4 The <code>CountingPtr</code> Template	383
20.2.5 A Simple Noninvasive Counter	386
20.2.6 A Simple Invasive Counter Template	388
20.2.7 Constness	390
20.2.8 Implicit Conversions	390
20.2.9 Comparisons	393
20.3 Afternotes	394
21 Tuples	395
21.1 Duos	395
21.2 Recursive Duos	401
21.2.1 Number of Fields	401

Contents	xiii
21.2.2 Type of Fields	403
21.2.3 Value of Fields	404
21.3 Tuple Construction	410
21.4 Afternotes	415
22 Function Objects and Callbacks	417
22.1 Direct, Indirect, and Inline Calls	418
22.2 Pointers and References to Functions	421
22.3 Pointer-to-Member Functions	423
22.4 Class Type Functors	426
22.4.1 A First Example of Class Type Functors	426
22.4.2 Type of Class Type Functors	428
22.5 Specifying Functors	429
22.5.1 Functors as Template Type Arguments	429
22.5.2 Functors as Function Call Arguments	430
22.5.3 Combining Function Call Parameters and Template Type Parameters	431
22.5.4 Functors as Nontype Template Arguments	432
22.5.5 Function Pointer Encapsulation	433
22.6 Introspection	436
22.6.1 Analyzing a Functor Type	436
22.6.2 Accessing Parameter Types	437
22.6.3 Encapsulating Function Pointers	439
22.7 Function Object Composition	445
22.7.1 Simple Composition	446
22.7.2 Mixed Type Composition	450
22.7.3 Reducing the Number of Parameters	454
22.8 Value Binders	457
22.8.1 Selecting the Binding	458
22.8.2 Bound Signature	460
22.8.3 Argument Selection	462
22.8.4 Convenience Functions	468
22.9 Functor Operations: A Complete Implementation	471
22.10 Afternotes	474

Appendixes	475
A The One-Definition Rule	475
A.1 Translation Units	475
A.2 Declarations and Definitions	476
A.3 The One-Definition Rule in Detail	477
A.3.1 One-per-Program Constraints	477
A.3.2 One-per-Translation Unit Constraints	479
A.3.3 Cross-Translation Unit Equivalence Constraints	481
B Overload Resolution	487
B.1 When Does Overload Resolution Kick In?	488
B.2 Simplified Overload Resolution	488
B.2.1 The Implied Argument for Member Functions	490
B.2.2 Refining the Perfect Match	492
B.3 Overloading Details	493
B.3.1 Prefer Nontemplates	493
B.3.2 Conversion Sequences	494
B.3.3 Pointer Conversions	494
B.3.4 Functors and Surrogate Functions	496
B.3.5 Other Overloading Contexts	497
Bibliography	499
Newsgroups	499
Books and Web Sites	500
Glossary	507
Index	517