

Contents

Preface	xvii
Aims	xvii
Subject Matter	xviii
Structure	xix
Supplementary Material	xx
Acknowledgments	xxi
Parachutes: Coda	xxii
About the Author	xxiii
Prologue	xxiv
A Dichotomy of Character	xxiv
Principles of UNIX Programming	xxv
Seven Signs of Successful C++ Software Libraries	xxvi
Balancing the Signs: Satisfaction, Dialecticism, and Idioms Old and New	xxxii
Example Libraries	xxxiii
Presentation Conventions	xxxvii
Fonts	xxxvii
. . . versus . . .	xxxvii
End Iterator Precomputation	xxxviii
Nested Class Type Qualification	xxxix
<i>NULL</i>	xxxix
Template Parameter Names	xl
Member and Namespace-Scope Type Names	xl
Calling Conventions	xl
Endpoint Iterators	xl
Namespace for Standard C Names	xl
Class Adaptors and Instance Adaptors	xl
Header File Names	xli
PART ONE Foundations	1
Chapter 1 The Standard Template Library	3
1.1 Core Concepts	3
1.2 Containers	4
1.3 Iterators	5
1.4 Algorithms	12
1.5 Function Objects	13
1.6 Allocators	13

Chapter 2	Extended STL Concepts, or When STL Meets the Real World	14
2.1	Terminology	14
2.2	Collections	15
2.3	Iterators	18
Chapter 3	Element Reference Categories	21
3.1	Introduction	21
3.2	C++ References	21
3.3	A Taxonomy of Element Reference Categories	23
3.4	Using Element Reference Categories	29
3.5	Defining operator <code>->()</code>	31
3.6	Element Reference Categories: Coda	31
Chapter 4	The Curious Untemporary Reference	32
Chapter 5	The <i>DRY SPOT</i> Principle	34
5.1	<i>DRY SPOTs</i> in C++	34
5.2	Not Quite <i>DRY SPOTs</i> in C++	36
5.3	Closed Namespaces	38
Chapter 6	The Law of Leaky Abstractions	40
Chapter 7	Contract Programming	42
7.1	Enforcement Types	42
7.2	Enforcement Mechanisms	43
Chapter 8	Constraints	45
8.1	Type System Leverage	45
8.2	Static Assertions	46
Chapter 9	Shims	48
9.1	Introduction	48
9.2	Primary Shims	49
9.3	Composite Shims	52
Chapter 10	Duck and Goose, or the Whimsical Bases of Partial Structural Conformance	57
10.1	Conformance	57
10.2	Explicit Semantic Conformance	62
10.3	Intersecting Conformance	64
Chapter 11	RAII	65
11.1	Mutability	65
11.2	Resource Source	65

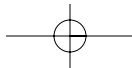
Contents	xi
Chapter 12 Template Tools	67
12.1 Traits	67
12.2 Type Generators	75
12.3 True Typedefs	76
Chapter 13 Inferred Interface Adaptation: Compile-Time Adaptation of Interface-Incomplete Types	77
13.1 Introduction	77
13.2 Adapting Interface-Incomplete Types	78
13.3 Adapting Immutable Collections	79
13.4 Inferred Interface Adaptation	80
13.5 Applying IIA to the Range	85
Chapter 14 Henney's Hypothesis, or When Templates Attack!	87
Chapter 15 The Independent Autonomies of <code>equal()</code> Friends	89
15.1 Beware Nonmember Friend Function Abuse	89
15.2 Collections and Their Iterators	91
Chapter 16 Essential Components	93
16.1 Introduction	93
16.2 <code>auto_buffer</code>	93
16.3 <code>filesystem_traits</code>	97
16.4 <code>file_path_buffer</code>	103
16.5 <code>scoped_handle</code>	107
16.6 <code>dl_call()</code>	109
PART TWO Collections	111
Chapter 17 Adapting the <code>glob</code> API	115
17.1 Introduction	115
17.2 Decomposition of the Longhand Version	119
17.3 <code>unixstl::glob_sequence</code>	121
17.4 Decomposition of the Shorthand Version	135
17.5 Summary	135
Chapter 18 Intermezzo: Constructor Clashes and Design That Is, If Not Bad, At Least Ill-Conceived for Seamless Evolution	137
Chapter 19 Adapting the <code>opendir/readdir</code> API	140
19.1 Introduction	140
19.2 Decomposition of the Longhand Version	142
19.3 <code>unixstl::readdir_sequence</code>	144
19.4 Alternate Implementations	161
19.5 Summary	162

Chapter 20	Adapting the FindFirstFile/FindNextFile API	164
20.1	Introduction	164
20.2	Decomposition of Examples	169
20.3	Sequence Design	171
20.4	winstl::basic_findfile_sequence	172
20.5	winstl::basic_findfile_sequence_const_iterator	178
20.6	winstl::basic_findfile_sequence_value_type	191
20.7	Shims	193
20.8	What, No Shims and Constructor Templates?	194
20.9	Summary	194
20.10	File System Enumeration with recls: Coda	195
Chapter 21	Intermezzo: When the Efficiency/Usability Balance Is Tipped: Enumerating FTP Server Directories	196
21.1	inetstl::basic_findfile_sequence	197
21.2	inetstl::basic_ftpdir_sequence	199
Chapter 22	Enumerating Processes and Modules	201
22.1	Collection Characteristics	202
22.2	winstl::pid_sequence	202
22.3	winstl::process_module_sequence	206
22.4	Enumerating All Modules on a System	206
22.5	Avoiding the System Pseudo Processes	208
22.6	Handling Optional API Headers	210
22.7	Summary	211
Chapter 23	The Fibonacci Sequence	212
23.1	Introduction	212
23.2	The Fibonacci Sequence	212
23.3	Fibonacci as an STL Sequence	212
23.4	Discoverability Failure	218
23.5	Defining Finite Bounds	218
23.6	Summary	225
Chapter 24	Adapting MFC's CArray Container Family	227
24.1	Introduction	227
24.2	Motivation	227
24.3	Emulating std::vector	230
24.4	Design Considerations	232
24.5	mfcstl::CArray_adaptor_base Interface	237
24.6	mfcstl::CArray_cadaptor	239
24.7	mfcstl::CArray_iadaptor	244
24.8	Construction	245
24.9	Allocator	245
24.10	Element Access Methods	246
24.11	Iteration	246
24.12	Size	248

Contents	xiii
24.13 Capacity	252
24.14 Comparison	253
24.15 Modifiers	256
24.16 Assignment and <code>swap()</code>	261
24.17 Summary	264
24.18 On the CD	265
Chapter 25 A Map of the Environment	266
25.1 Introduction	266
25.2 Motivation	266
25.3 <code>getenv()</code> , <code>putenv()</code> , <code>setenv()/unsetenv()</code> , and <code>environ</code>	267
25.4 <code>platformstl::environment_variable_traits</code>	268
25.5 Planning the Interface	271
25.6 Lookup by Name	271
25.7 Inserting, Updating, and Deleting Values by Name	277
25.8 Iteration	278
25.9 Final Iteration Implementation	286
25.10 Heterogeneous Reference Categories?	297
25.11 <code>size()</code> and Subscript by Index	297
25.12 Summary	298
25.13 On the CD	298
Chapter 26 Traveling Back and Forth on the Z-Plane	299
26.1 Prologue	299
26.2 Introduction	299
26.3 Version 1: Forward Iteration	302
26.4 Version 2: Bidirectional Iteration	304
26.5 Handling External Change	306
26.6 <code>winstl::child_window_sequence</code>	309
26.7 Bidirectional Iterator Blues	309
26.8 <code>winstl::zorder_iterator</code> : A Reversal of Self	315
26.9 Finalizing the Window Peer Sequences	321
26.10 Summary	323
26.11 Z-Plane: Coda	323
Chapter 27 String Tokenization	324
27.1 Introduction	324
27.2 <code>strtok()</code>	325
27.3 <code>SynesisSTL::StringTokeniser</code>	327
27.4 Tokenization Use Cases	329
27.5 Other Tokenization Alternatives	330
27.6 <code>stlsoft::string_tokeniser</code>	331
27.7 Test Drive	340
27.8 The Policy Folly	345
27.9 Performance	348
27.10 Summary	352

Chapter 28	Adapting COM Enumerators	353
28.1	Introduction	353
28.2	Motivation	353
28.3	COM Enumerators	355
28.4	Decomposition of the Longhand Version	358
28.5	<code>comstl::enumerator_sequence</code>	359
28.6	<code>comstl::enumerator_sequence::iterator</code>	368
28.7	<code>comstl::enumerator_sequence::iterator::enumeration_context</code>	372
28.8	Iterator Cloning Policies	381
28.9	Choosing a Default Cloning Policy: Applying the <i>Principle of Least Surprise</i>	385
28.10	Summary	390
28.11	Coming Next	391
Chapter 29	Intermezzo: Correcting Minor Design Omissions with Member Type Inference	392
Chapter 30	Adapting COM Collections	394
30.1	Introduction	394
30.2	Motivation	394
30.3	<code>comstl::collection_sequence</code>	398
30.4	Enumerator Acquisition Policies	403
30.5	Summary	406
Chapter 31	Gathering Scattered I/O	407
31.1	Introduction	407
31.2	Scatter/Gather I/O	407
31.3	Scatter/Gather I/O APIs	409
31.4	Adapting <code>ACE_Message_Queue</code>	414
31.5	Time for Some Cake	420
31.6	Summary	427
Chapter 32	Argument-Dependent Return-Type Variance	428
32.1	Introduction	428
32.2	Borrowing a Jewel from Ruby	428
32.3	Dual-Semantic Subscripting in C++	430
32.4	Generalized Compatibility via String Access Shims	431
32.5	A Fly in the <code>int</code> -ment	432
32.6	Selecting Return Type and Overload	433
32.7	Summary	434
Chapter 33	External Iterator Invalidation	435
33.1	Element-Interface Coherence	435
33.2	Windows <code>Listbox</code> and <code>ComboBox</code> Controls	437
33.3	Enumerating Registry Keys and Values	444
33.4	Summary	463
33.5	On the CD	463

Contents	xv
PART THREE Iterators	465
Chapter 34 An Enhanced ostream_iterator	467
34.1 Introduction	467
34.2 std::ostream_iterator	469
34.3 stlsoft::ostream_iterator	470
34.4 Defining Stream Insertion Operators	475
34.5 Summary	476
Chapter 35 Intermezzo: Proscribing Fatuous Output Iterator Syntax Using the Dereference Proxy Pattern	477
35.1 stlsoft::ostream_iterator::deref_proxy	478
Chapter 36 Transform Iterator	481
36.1 Introduction	481
36.2 Motivation	482
36.3 Defining Iterator Adaptors	485
36.4 stlsoft::transform_iterator	487
36.5 Composite Transformations	496
36.6 <i>DRY SPOT</i> Violations?	497
36.7 A Spoonful of Sequence Helps the Medicine . . . ?	501
36.8 Summary	501
36.9 On the CD	502
Chapter 37 Intermezzo: Discretion Being the Better Part of Nomenclature . . .	503
Chapter 38 Member Selector Iterator	506
38.1 Introduction	506
38.2 Motivation	506
38.3 stlsoft::member_selector_iterator	509
38.4 Creator Function Woes	511
38.5 Summary	518
38.6 On the CD	518
Chapter 39 C-Style String Concatenation	519
39.1 Motivation	519
39.2 An Inflexible Version	520
39.3 stlsoft::cstring_concatenator_iterator	522
39.4 Creator Functions	524
39.5 Summary	525
39.6 On the CD	526
Chapter 40 String Object Concatenation	527
40.1 Introduction	527
40.2 stlsoft::string_concatenator_iterator	527
40.3 Heterogeneity of String Types	530



xvi	Contents
40.4 But . . .	530
40.5 Summary	532
Chapter 41 Adapted Iterators Traits	533
41.1 Introduction	533
41.2 <code>stlsoft::adapted_iterator_traits</code>	533
41.3 Summary	540
41.4 On the CD	541
Chapter 42 Filtered Iteration	542
42.1 Introduction	542
42.2 An Invalid Version	542
42.3 Member Iterators Define the Range	543
42.4 So . . . ?	544
42.5 <code>stlsoft::filter_iterator</code>	545
42.6 Constraining the Iterator Category	549
42.7 Summary	550
42.8 On the CD	550
Chapter 43 Composite Iterator Adaptations	551
43.1 Introduction	551
43.2 Transforming a Filtered Iterator	551
43.3 Filtering a Transformed Iterator	553
43.4 Hedging Our Bets	554
43.5 Summary	554
Epilogue	555
Bibliography	556
Index	559

