

Contents

Preface to the Second Edition	xxiii
Acknowledgments for the Second Edition	xxiv
Preface to the First Edition	xxv
Acknowledgments for the First Edition	xxvi
1 About This Book	1
1.1 Why This Book	1
1.2 Before Reading This Book	2
1.3 Style and Structure of the Book	2
1.4 How to Read This Book	4
1.5 State of the Art	5
1.6 Example Code and Additional Information	5
1.7 Feedback	5
2 Introduction to C++ and the Standard Library	7
2.1 History of the C++ Standards	7
2.1.1 Common Questions about the C++11 Standard	8
2.1.2 Compatibility between C++98 and C++11	9
2.2 Complexity and Big-O Notation	10
3 New Language Features	13
3.1 New C++11 Language Features	13
3.1.1 Important Minor Syntax Cleanups	13
3.1.2 Automatic Type Deduction with <code>auto</code>	14
3.1.3 Uniform Initialization and Initializer Lists	15
3.1.4 Range-Based <code>for</code> Loops	17
3.1.5 Move Semantics and Rvalue References	19

3.1.6	New String Literals	23
3.1.7	Keyword <code>noexcept</code>	24
3.1.8	Keyword <code>constexpr</code>	26
3.1.9	New Template Features	26
3.1.10	Lambdas	28
3.1.11	Keyword <code>decltype</code>	32
3.1.12	New Function Declaration Syntax	32
3.1.13	Scoped Enumerations	32
3.1.14	New Fundamental Data Types	33
3.2	Old “New” Language Features	33
3.2.1	Explicit Initialization for Fundamental Types	37
3.2.2	Definition of <code>main()</code>	37
4	General Concepts	39
4.1	Namespace <code>std</code>	39
4.2	Header Files	40
4.3	Error and Exception Handling	41
4.3.1	Standard Exception Classes	41
4.3.2	Members of Exception Classes	44
4.3.3	Passing Exceptions with Class <code>exception_ptr</code>	52
4.3.4	Throwing Standard Exceptions	53
4.3.5	Deriving from Standard Exception Classes	54
4.4	Callable Objects	54
4.5	Concurrency and Multithreading	55
4.6	Allocators	57
5	Utilities	59
5.1	Pairs and Tuples	60
5.1.1	Pairs	60
5.1.2	Tuples	68
5.1.3	I/O for Tuples	74
5.1.4	Conversions between <code>tuples</code> and <code>pairs</code>	75
5.2	Smart Pointers	76
5.2.1	Class <code>shared_ptr</code>	76
5.2.2	Class <code>weak_ptr</code>	84
5.2.3	Misusing Shared Pointers	89
5.2.4	Shared and Weak Pointers in Detail	92
5.2.5	Class <code>unique_ptr</code>	98

5.2.6	Class <code>unique_ptr</code> in Detail	110
5.2.7	Class <code>auto_ptr</code>	113
5.2.8	Final Words on Smart Pointers	114
5.3	Numeric Limits	115
5.4	Type Traits and Type Utilities	122
5.4.1	Purpose of Type Traits	122
5.4.2	Type Traits in Detail	125
5.4.3	Reference Wrappers	132
5.4.4	Function Type Wrappers	133
5.5	Auxiliary Functions	134
5.5.1	Processing the Minimum and Maximum	134
5.5.2	Swapping Two Values	136
5.5.3	Supplementary Comparison Operators	138
5.6	Compile-Time Fractional Arithmetic with Class <code>ratio<></code>	140
5.7	Clocks and Timers	143
5.7.1	Overview of the Chrono Library	143
5.7.2	Durations	144
5.7.3	Clocks and Timepoints	149
5.7.4	Date and Time Functions by C and POSIX	157
5.7.5	Blocking with Timers	160
5.8	Header Files <code><cstdlib></code> , <code><stdlib.h></code> , and <code><cstring></code>	161
5.8.1	Definitions in <code><cstdlib></code>	161
5.8.2	Definitions in <code><stdlib.h></code>	162
5.8.3	Definitions in <code><cstring></code>	163
6	The Standard Template Library	165
6.1	STL Components	165
6.2	Containers	167
6.2.1	Sequence Containers	169
6.2.2	Associative Containers	177
6.2.3	Unordered Containers	180
6.2.4	Associative Arrays	185
6.2.5	Other Containers	187
6.2.6	Container Adapters	188
6.3	Iterators	188
6.3.1	Further Examples of Using Associative and Unordered Containers	193
6.3.2	Iterator Categories	198

6.4	Algorithms	199
6.4.1	Ranges	203
6.4.2	Handling Multiple Ranges	207
6.5	Iterator Adapters	210
6.5.1	Insert Iterators	210
6.5.2	Stream Iterators	212
6.5.3	Reverse Iterators	214
6.5.4	Move Iterators	216
6.6	User-Defined Generic Functions	216
6.7	Manipulating Algorithms	217
6.7.1	“Removing” Elements	218
6.7.2	Manipulating Associative and Unordered Containers	221
6.7.3	Algorithms versus Member Functions	223
6.8	Functions as Algorithm Arguments	224
6.8.1	Using Functions as Algorithm Arguments	224
6.8.2	Predicates	226
6.9	Using Lambdas	229
6.10	Function Objects	233
6.10.1	Definition of Function Objects	233
6.10.2	Predefined Function Objects	239
6.10.3	Binders	241
6.10.4	Function Objects and Binders versus Lambdas	243
6.11	Container Elements	244
6.11.1	Requirements for Container Elements	244
6.11.2	Value Semantics or Reference Semantics	245
6.12	Errors and Exceptions inside the STL	245
6.12.1	Error Handling	246
6.12.2	Exception Handling	248
6.13	Extending the STL	250
6.13.1	Integrating Additional Types	250
6.13.2	Deriving from STL Types	251
7	STL Containers	253
7.1	Common Container Abilities and Operations	254
7.1.1	Container Abilities	254
7.1.2	Container Operations	254
7.1.3	Container Types	260

7.2	Arrays	261
7.2.1	Abilities of Arrays	261
7.2.2	Array Operations	263
7.2.3	Using arrays as C-Style Arrays	267
7.2.4	Exception Handling	268
7.2.5	Tuple Interface	268
7.2.6	Examples of Using Arrays	268
7.3	Vectors	270
7.3.1	Abilities of Vectors	270
7.3.2	Vector Operations	273
7.3.3	Using Vectors as C-Style Arrays	278
7.3.4	Exception Handling	278
7.3.5	Examples of Using Vectors	279
7.3.6	Class <code>vector<bool></code>	281
7.4	Deque	283
7.4.1	Abilities of Deques	284
7.4.2	Deque Operations	285
7.4.3	Exception Handling	288
7.4.4	Examples of Using Deques	288
7.5	Lists	290
7.5.1	Abilities of Lists	290
7.5.2	List Operations	291
7.5.3	Exception Handling	296
7.5.4	Examples of Using Lists	298
7.6	Forward Lists	300
7.6.1	Abilities of Forward Lists	300
7.6.2	Forward List Operations	302
7.6.3	Exception Handling	311
7.6.4	Examples of Using Forward Lists	312
7.7	Sets and Multisets	314
7.7.1	Abilities of Sets and Multisets	315
7.7.2	Set and Multiset Operations	316
7.7.3	Exception Handling	325
7.7.4	Examples of Using Sets and Multisets	325
7.7.5	Example of Specifying the Sorting Criterion at Runtime	328

7.8	Maps and Multimaps	331
7.8.1	Abilities of Maps and Multimaps	332
7.8.2	Map and Multimap Operations	333
7.8.3	Using Maps as Associative Arrays	343
7.8.4	Exception Handling	345
7.8.5	Examples of Using Maps and Multimaps	345
7.8.6	Example with Maps, Strings, and Sorting Criterion at Runtime	351
7.9	Unordered Containers	355
7.9.1	Abilities of Unordered Containers	357
7.9.2	Creating and Controlling Unordered Containers	359
7.9.3	Other Operations for Unordered Containers	367
7.9.4	The Bucket Interface	374
7.9.5	Using Unordered Maps as Associative Arrays	374
7.9.6	Exception Handling	375
7.9.7	Examples of Using Unordered Containers	375
7.10	Other STL Containers	385
7.10.1	Strings as STL Containers	385
7.10.2	Ordinary C-Style Arrays as STL Containers	386
7.11	Implementing Reference Semantics	388
7.12	When to Use Which Container	392
8	STL Container Members in Detail	397
8.1	Type Definitions	397
8.2	Create, Copy, and Destroy Operations	400
8.3	Nonmodifying Operations	403
8.3.1	Size Operations	403
8.3.2	Comparison Operations	404
8.3.3	Nonmodifying Operations for Associative and Unordered Containers	404
8.4	Assignments	406
8.5	Direct Element Access	408
8.6	Operations to Generate Iterators	410
8.7	Inserting and Removing Elements	411
8.7.1	Inserting Single Elements	411
8.7.2	Inserting Multiple Elements	416
8.7.3	Removing Elements	417
8.7.4	Resizing	420

8.8	Special Member Functions for Lists and Forward Lists	420
8.8.1	Special Member Functions for Lists (and Forward Lists)	420
8.8.2	Special Member Functions for Forward Lists Only	423
8.9	Container Policy Interfaces	427
8.9.1	Nonmodifying Policy Functions	427
8.9.2	Modifying Policy Functions	428
8.9.3	Bucket Interface for Unordered Containers	429
8.10	Allocator Support	430
8.10.1	Fundamental Allocator Members	430
8.10.2	Constructors with Optional Allocator Parameters	430
9	STL Iterators	433
9.1	Header Files for Iterators	433
9.2	Iterator Categories	433
9.2.1	Output Iterators	433
9.2.2	Input Iterators	435
9.2.3	Forward Iterators	436
9.2.4	Bidirectional Iterators	437
9.2.5	Random-Access Iterators	438
9.2.6	The Increment and Decrement Problem of Vector Iterators	440
9.3	Auxiliary Iterator Functions	441
9.3.1	<code>advance()</code>	441
9.3.2	<code>next()</code> and <code>prev()</code>	443
9.3.3	<code>distance()</code>	445
9.3.4	<code>iter_swap()</code>	446
9.4	Iterator Adapters	448
9.4.1	Reverse Iterators	448
9.4.2	Insert Iterators	454
9.4.3	Stream Iterators	460
9.4.4	Move Iterators	466
9.5	Iterator Traits	466
9.5.1	Writing Generic Functions for Iterators	468
9.6	Writing User-Defined Iterators	471

10 STL Function Objects and Using Lambdas	475
10.1 The Concept of Function Objects	475
10.1.1 Function Objects as Sorting Criteria	476
10.1.2 Function Objects with Internal State	478
10.1.3 The Return Value of <code>for_each()</code>	482
10.1.4 Predicates versus Function Objects	483
10.2 Predefined Function Objects and Binders	486
10.2.1 Predefined Function Objects	486
10.2.2 Function Adapters and Binders	487
10.2.3 User-Defined Function Objects for Function Adapters	495
10.2.4 Deprecated Function Adapters	497
10.3 Using Lambdas	499
10.3.1 Lambdas versus Binders	499
10.3.2 Lambdas versus Stateful Function Objects	500
10.3.3 Lambdas Calling Global and Member Functions	502
10.3.4 Lambdas as Hash Function, Sorting, or Equivalence Criterion	504
11 STL Algorithms	505
11.1 Algorithm Header Files	505
11.2 Algorithm Overview	505
11.2.1 A Brief Introduction	506
11.2.2 Classification of Algorithms	506
11.3 Auxiliary Functions	517
11.4 The <code>for_each()</code> Algorithm	519
11.5 Nonmodifying Algorithms	524
11.5.1 Counting Elements	524
11.5.2 Minimum and Maximum	525
11.5.3 Searching Elements	528
11.5.4 Comparing Ranges	542
11.5.5 Predicates for Ranges	550
11.6 Modifying Algorithms	557
11.6.1 Copying Elements	557
11.6.2 Moving Elements	561
11.6.3 Transforming and Combining Elements	563
11.6.4 Swapping Elements	566
11.6.5 Assigning New Values	568
11.6.6 Replacing Elements	571

11.7	Removing Algorithms	575
11.7.1	Removing Certain Values	575
11.7.2	Removing Duplicates	578
11.8	Mutating Algorithms	583
11.8.1	Reversing the Order of Elements	583
11.8.2	Rotating Elements	584
11.8.3	Permuting Elements	587
11.8.4	Shuffling Elements	589
11.8.5	Moving Elements to the Front	592
11.8.6	Partition into Two Subranges	594
11.9	Sorting Algorithms	596
11.9.1	Sorting All Elements	596
11.9.2	Partial Sorting	599
11.9.3	Sorting According to the n th Element	602
11.9.4	Heap Algorithms	604
11.10	Sorted-Range Algorithms	608
11.10.1	Searching Elements	608
11.10.2	Merging Elements	614
11.11	Numeric Algorithms	623
11.11.1	Processing Results	623
11.11.2	Converting Relative and Absolute Values	627
12	Special Containers	631
12.1	Stacks	632
12.1.1	The Core Interface	633
12.1.2	Example of Using Stacks	633
12.1.3	A User-Defined Stack Class	635
12.1.4	Class <code>stack<></code> in Detail	637
12.2	Queues	638
12.2.1	The Core Interface	639
12.2.2	Example of Using Queues	640
12.2.3	A User-Defined Queue Class	641
12.2.4	Class <code>queue<></code> in Detail	641
12.3	Priority Queues	641
12.3.1	The Core Interface	643
12.3.2	Example of Using Priority Queues	643
12.3.3	Class <code>priority_queue<></code> in Detail	644

12.4	Container Adapters in Detail	645
12.4.1	Type Definitions	645
12.4.2	Constructors	646
12.4.3	Supplementary Constructors for Priority Queues	646
12.4.4	Operations	647
12.5	Bitsets	650
12.5.1	Examples of Using Bitsets	651
12.5.2	Class <code>bitset</code> in Detail	653
13	Strings	655
13.1	Purpose of the String Classes	656
13.1.1	A First Example: Extracting a Temporary Filename	656
13.1.2	A Second Example: Extracting Words and Printing Them Backward	660
13.2	Description of the String Classes	663
13.2.1	String Types	663
13.2.2	Operation Overview	666
13.2.3	Constructors and Destructor	667
13.2.4	Strings and C-Strings	668
13.2.5	Size and Capacity	669
13.2.6	Element Access	671
13.2.7	Comparisons	672
13.2.8	Modifiers	673
13.2.9	Substrings and String Concatenation	676
13.2.10	Input/Output Operators	677
13.2.11	Searching and Finding	678
13.2.12	The Value <code>npos</code>	680
13.2.13	Numeric Conversions	681
13.2.14	Iterator Support for Strings	684
13.2.15	Internationalization	689
13.2.16	Performance	692
13.2.17	Strings and Vectors	692
13.3	String Class in Detail	693
13.3.1	Type Definitions and Static Values	693
13.3.2	Create, Copy, and Destroy Operations	694
13.3.3	Operations for Size and Capacity	696
13.3.4	Comparisons	697
13.3.5	Character Access	699
13.3.6	Generating C-Strings and Character Arrays	700

13.3.7	Modifying Operations	700
13.3.8	Searching and Finding	708
13.3.9	Substrings and String Concatenation	711
13.3.10	Input/Output Functions	712
13.3.11	Numeric Conversions	713
13.3.12	Generating Iterators	714
13.3.13	Allocator Support	715
14	Regular Expressions	717
14.1	The Regex Match and Search Interface	717
14.2	Dealing with Subexpressions	720
14.3	Regex Iterators	726
14.4	Regex Token Iterators	727
14.5	Replacing Regular Expressions	730
14.6	Regex Flags	732
14.7	Regex Exceptions	735
14.8	The Regex ECMAScript Grammar	738
14.9	Other Grammars	739
14.10	Basic Regex Signatures in Detail	740
15	Input/Output Using Stream Classes	743
15.1	Common Background of I/O Streams	744
15.1.1	Stream Objects	744
15.1.2	Stream Classes	744
15.1.3	Global Stream Objects	745
15.1.4	Stream Operators	745
15.1.5	Manipulators	746
15.1.6	A Simple Example	746
15.2	Fundamental Stream Classes and Objects	748
15.2.1	Classes and Class Hierarchy	748
15.2.2	Global Stream Objects	751
15.2.3	Header Files	752
15.3	Standard Stream Operators << and >>	753
15.3.1	Output Operator <<	753
15.3.2	Input Operator >>	754
15.3.3	Input/Output of Special Types	755

15.4	State of Streams	758
15.4.1	Constants for the State of Streams	758
15.4.2	Member Functions Accessing the State of Streams	759
15.4.3	Stream State and Boolean Conditions	760
15.4.4	Stream State and Exceptions	762
15.5	Standard Input/Output Functions	767
15.5.1	Member Functions for Input	768
15.5.2	Member Functions for Output	771
15.5.3	Example Uses	772
15.5.4	sentry Objects	772
15.6	Manipulators	774
15.6.1	Overview of All Manipulators	774
15.6.2	How Manipulators Work	776
15.6.3	User-Defined Manipulators	777
15.7	Formatting	779
15.7.1	Format Flags	779
15.7.2	Input/Output Format of Boolean Values	781
15.7.3	Field Width, Fill Character, and Adjustment	781
15.7.4	Positive Sign and Uppercase Letters	784
15.7.5	Numeric Base	785
15.7.6	Floating-Point Notation	787
15.7.7	General Formatting Definitions	789
15.8	Internationalization	790
15.9	File Access	791
15.9.1	File Stream Classes	791
15.9.2	Rvalue and Move Semantics for File Streams	795
15.9.3	File Flags	796
15.9.4	Random Access	799
15.9.5	Using File Descriptors	801
15.10	Stream Classes for Strings	802
15.10.1	String Stream Classes	802
15.10.2	Move Semantics for String Streams	806
15.10.3	char* Stream Classes	807
15.11	Input/Output Operators for User-Defined Types	810
15.11.1	Implementing Output Operators	810
15.11.2	Implementing Input Operators	812
15.11.3	Input/Output Using Auxiliary Functions	814

15.11.4	User-Defined Format Flags	815
15.11.5	Conventions for User-Defined Input/Output Operators	818
15.12	Connecting Input and Output Streams	819
15.12.1	Loose Coupling Using <code>tie()</code>	819
15.12.2	Tight Coupling Using Stream Buffers	820
15.12.3	Redirecting Standard Streams	822
15.12.4	Streams for Reading and Writing	824
15.13	The Stream Buffer Classes	826
15.13.1	The Stream Buffer Interfaces	826
15.13.2	Stream Buffer Iterators	828
15.13.3	User-Defined Stream Buffers	832
15.14	Performance Issues	844
15.14.1	Synchronization with C's Standard Streams	845
15.14.2	Buffering in Stream Buffers	845
15.14.3	Using Stream Buffers Directly	846
16	Internationalization	849
16.1	Character Encodings and Character Sets	850
16.1.1	Multibyte and Wide-Character Text	850
16.1.2	Different Character Sets	851
16.1.3	Dealing with Character Sets in C++	852
16.1.4	Character Traits	853
16.1.5	Internationalization of Special Characters	857
16.2	The Concept of Locales	857
16.2.1	Using Locales	858
16.2.2	Locale Facets	864
16.3	Locales in Detail	866
16.4	Facets in Detail	869
16.4.1	Numeric Formatting	870
16.4.2	Monetary Formatting	874
16.4.3	Time and Date Formatting	884
16.4.4	Character Classification and Conversion	891
16.4.5	String Collation	904
16.4.6	Internationalized Messages	905

17 Numerics	907
17.1 Random Numbers and Distributions	907
17.1.1 A First Example	908
17.1.2 Engines	912
17.1.3 Engines in Detail	915
17.1.4 Distributions	917
17.1.5 Distributions in Detail	921
17.2 Complex Numbers	925
17.2.1 Class <code>complex<></code> in General	925
17.2.2 Examples Using Class <code>complex<></code>	926
17.2.3 Operations for Complex Numbers	928
17.2.4 Class <code>complex<></code> in Detail	935
17.3 Global Numeric Functions	941
17.4 Valarrays	943
18 Concurrency	945
18.1 The High-Level Interface: <code>async()</code> and Futures	946
18.1.1 A First Example Using <code>async()</code> and Futures	946
18.1.2 An Example of Waiting for Two Tasks	955
18.1.3 Shared Futures	960
18.2 The Low-Level Interface: Threads and Promises	964
18.2.1 Class <code>std::thread</code>	964
18.2.2 Promises	969
18.2.3 Class <code>packaged_task<></code>	972
18.3 Starting a Thread in Detail	973
18.3.1 <code>async()</code> in Detail	974
18.3.2 Futures in Detail	975
18.3.3 Shared Futures in Detail	976
18.3.4 Class <code>std::promise</code> in Detail	977
18.3.5 Class <code>std::packaged_task</code> in Detail	977
18.3.6 Class <code>std::thread</code> in Detail	979
18.3.7 Namespace <code>this_thread</code>	981
18.4 Synchronizing Threads, or the Problem of Concurrency	982
18.4.1 Beware of Concurrency!	982
18.4.2 The Reason for the Problem of Concurrent Data Access	983
18.4.3 What Exactly Can Go Wrong (the Extent of the Problem)	983
18.4.4 The Features to Solve the Problems	987

18.5	Mutexes and Locks	989
18.5.1	Using Mutexes and Locks	989
18.5.2	Mutexes and Locks in Detail	998
18.5.3	Calling Once for Multiple Threads	1000
18.6	Condition Variables	1003
18.6.1	Purpose of Condition Variables	1003
18.6.2	A First Complete Example for Condition Variables	1004
18.6.3	Using Condition Variables to Implement a Queue for Multiple Threads	1006
18.6.4	Condition Variables in Detail	1009
18.7	Atomics	1012
18.7.1	Example of Using Atomics	1012
18.7.2	Atomics and Their High-Level Interface in Detail	1016
18.7.3	The C-Style Interface of Atomics	1019
18.7.4	The Low-Level Interface of Atomics	1019
19	Allocators	1023
19.1	Using Allocators as an Application Programmer	1023
19.2	A User-Defined Allocator	1024
19.3	Using Allocators as a Library Programmer	1026
	Bibliography	1031
	Newsgroups and Forums	1031
	Books and Web Sites	1032
	Index	1037

Preface to the Second Edition

I never thought that the first edition of this book would sell so long. But now, after twelve years, it's time for a new edition that covers C++11, the new C++ standard.

Note that this means more than simply adding new libraries. C++ has changed. Almost all typical applications of parts of the library look a bit different now. This is not the result of a huge language change. It's the result of many minor changes, such as using rvalue references and move semantics, range-based `for` loops, `auto`, and new template features. Thus, besides presenting new libraries and supplementary features of existing libraries, almost all of the examples in this book were rewritten at least partially. Nevertheless, to support programmers who still use "old" C++ environments, this book will describe differences between C++ versions whenever they appear.

I learned C++11 the hard way. Because I didn't follow the standardization as it was happening I started to look at C++11 about two years ago. I really had trouble understanding it. But the people on the standardization committee helped me to describe and present the new features as they are intended to be used now.

Note, finally, that this book now has a problem: Although the book's size grew from about 800 to more than 1,100 pages, I still can't present the C++ standard library as a whole. The library part of the new C++11 standard alone now has about 750 pages, written in very condensed form without much explanation. For this reason, I had to decide which features to describe and in how much detail. Again, many people in the C++ community helped me to make this decision. The intent was to concentrate on what the average application programmer needs. For some missing parts, I provide a supplementary chapter on the Web site of this book, <http://www.cppstdlib.com>, but you still will find details not mentioned here in the standard.

The art of teaching is not the art of presenting everything. It's the art of separating the wheat from the chaff so that you get the most out of it. May the exercise succeed.

Acknowledgments for the Second Edition

This book presents ideas, concepts, solutions, and examples from many sources. Over the past several years, the C++ community introduced many ideas, concepts, proposals, and enhancements to C++ that became part of C++11. Thus, again I'd like to thank all the people who helped and supported me while preparing this new edition.

First, I'd like to thank everyone in the C++ community and on the C++ standardization committee. Besides all the work to add new language and library features, they had a hard time explaining everything to me, but they did so with patience and enthusiasm.

Scott Meyers and Anthony Williams allowed me to use their teaching material and book manuscripts so that I could find many useful examples not yet publicly available.

I'd also like to thank everyone who reviewed this book and gave valuable feedback and clarifications: Dave Abrahams, Alberto Ganesh Barbati, Pete Becker, Thomas Becker, Hans Boehm, Walter E. Brown, Paolo Carlini, Lawrence Crowl, Beman Dawes, Doug Gregor, David Grigsby, Pablo Halpern, Howard Hinnant, John Lakos, Bronek Kozicki, Dietmar Kühl, Daniel Krügler, Mat Marcus, Jens Maurer, Alisdair Meredith, Bartosz Milewski, P. J. Plauger, Tobias Schüle, Peter Sommerlad, Jonathan Wakely, and Anthony Williams.

There is one person who did an especially outstanding job. Whenever I had a question, Daniel Krügler answered almost immediately with incredible accurateness and knowledge. Everyone in the standardization process knows that he treats everybody this way. Without him, both the C++ standard and this book would not have the quality they have now.

Many thanks to my editor Peter Gordon, Kim Boedigheimer, John Fuller, and Anna Popick from Addison-Wesley. Besides their support, they found the right balance between patience and pressure. The copy editor Evelyn Pyle and the proofreader Diane Freed did an incredible job translating my German English into American English. In addition, thanks to Frank Mittelbach for solving my L^AT_EX issues.

Last but not least, all my thanks go to Jutta Eckstein. Jutta has the wonderful ability to force and support people in their ideals, ideas, and goals. While most people experience this only when working with her, I have the honor to benefit in my day-to-day life.

Preface to the First Edition

In the beginning, I only planned to write a small German book (400 pages or so) about the C++ standard library. That was in 1993. Now, in 1999 you see the result — a book in English with more than 800 pages of facts, figures, and examples. My goal is to describe the C++ standard library so that all (or almost all) your programming questions are answered before you think of the question. Note, however, that this is not a complete description of all aspects of the C++ standard library. Instead, I present the most important topics necessary for learning and programming in C++ by using its standard library.

Each topic is described based on the general concepts; this discussion then leads to the specific details needed to support everyday programming tasks. Specific code examples are provided to help you understand the concepts and the details.

That's it — in a nutshell. I hope you get as much pleasure from reading this book as I did from writing it. Enjoy!

Acknowledgments for the First Edition

This book presents ideas, concepts, solutions, and examples from many sources. In a way it does not seem fair that my name is the only name on the cover. Thus, I'd like to thank all the people and companies who helped and supported me during the past few years.

First, I'd like to thank Dietmar Kühl. Dietmar is an expert on C++, especially on input/output streams and internationalization (he implemented an I/O stream library just for fun). He not only translated major parts of this book from German to English, he also wrote sections of this book using his expertise. In addition, he provided me with invaluable feedback over the years.

Second, I'd like to thank all the reviewers and everyone else who gave me their opinion. These people endow the book with a quality it would never have had without their input. (Because the list is extensive, please forgive me for any oversight.) The reviewers for the English version of this book included Chuck Allison, Greg Comeau, James A. Crotinger, Gabriel Dos Reis, Alan Ezust, Nathan Myers, Werner Mossner, Todd Veldhuizen, Chichiang Wan, Judy Ward, and Thomas Wikehult. The German reviewers included Ralf Boecker, Dirk Herrmann, Dietmar Kühl, Edda Lörke, Herbert Scheubner, Dominik Strasser, and Martin Weitzel. Additional input was provided by Matt Austern, Valentin Bonnard, Greg Colvin, Beman Dawes, Bill Gibbons, Lois Goldthwaite, Andrew Koenig, Steve Rumsby, Bjarne Stroustrup, and David Vandevoorde.

Special thanks to Dave Abrahams, Janet Cocker, Catherine Ohala, and Maureen Willard who reviewed and edited the whole book very carefully. Their feedback was an incredible contribution to the quality of this book.

A special thanks goes to my “personal living dictionary” — Herb Sutter — the author of the famous “Guru of the Week” (a regular series of C++ programming problems that is published on the `comp.lang.c++.moderated` Internet newsgroup).

I'd also like to thank all the people and companies who gave me the opportunity to test my examples on different platforms with different compilers. Many thanks to Steve Adamczyk, Mike Anderson, and John Spicer from EDG for their great compiler and their support. It was a big help during the standardization process and the writing of this book. Many thanks to P. J. Plauger and Dinkumware, Ltd, for their early standard-conforming implementation of the C++ standard library. Many thanks to Andreas Hommel and Metrowerks for an evaluative version of their CodeWarrior Programming Environment. Many thanks to all the developers of the free GNU and egcs compilers. Many thanks to Microsoft for an evaluative version of Visual C++. Many thanks to Roland Hartinger

from Siemens Nixdorf Informations Systems AG for a test version of their C++ compiler. Many thanks to Topjects GmbH for an evaluative version of the ObjectSpace library implementation.

Many thanks to everyone from Addison Wesley Longman who worked with me. Among others this includes Janet Cocker, Mike Hendrickson, Debbie Lafferty, Marina Lang, Chanda Leary, Catherine Ohala, Marty Rabinowitz, Susanne Spitzer, and Maureen Willard. It was fun.

In addition, I'd like to thank the people at BREDEX GmbH and all the people in the C++ community, particularly those involved with the standardization process, for their support and patience (sometimes I ask really silly questions).

Last but not least, many thanks and kisses for my family: Ulli, Lucas, Anica, and Frederic. I definitely did not have enough time for them due to the writing of this book.

Have fun and be human!

Chapter 1

About This Book

1.1 Why This Book

Soon after its introduction, C++ became a de facto standard in object-oriented programming. This led to the goal of standardization. Only by having a standard could programs be written that would run on different platforms — from PCs to mainframes. Furthermore, a standard *library* would enable programmers to use general components and a higher level of abstraction without losing portability rather than having to develop all code from scratch.

Now, with the second standard, called C++11 (see Section 2.1, page 7, for the detailed history of C++ standards), we have a huge C++ standard library whose specification requires more than double the size of the core language features. The library enables the use of

- Input/output (I/O) classes
- String types and regular expressions
- Various data structures, such as dynamic arrays, linked lists, binary trees, and hash tables
- Various algorithms, such as a variety of sorting algorithms
- Classes for multithreading and concurrency
- Classes for internationalization support
- Numeric classes
- Plenty of utilities

However, the library is not self-explanatory. To use these components and to benefit from their power, you need an introduction that explains the concepts and the important details instead of simply listing the classes and their functions. This book is written exactly for that purpose. First, it introduces the library and all its components from a conceptual point of view. Next, the book describes the details needed for practical programming. Examples are included to demonstrate the exact use of the components. Thus, this book is a detailed introduction to the C++ library for both the beginner and the practicing programmer. Armed with the data provided herein, you should be able to take full advantage of the C++ standard library.

Caveat: I don't promise that everything described is easy and self-explanatory. The library provides a lot of flexibility, but flexibility for nontrivial purposes has a price. The library has traps and pitfalls, which I point out when we encounter them and suggest ways of avoiding them.

1.2 Before Reading This Book

To get the most from this book, you should already know C++. (The book describes the standard components of C++ but not the language itself.) You should be familiar with the concepts of classes, inheritance, templates, exception handling, and namespaces. However, you don't have to know all the minor details about the language. The important details are described in the book; the minor details about the language are more important for people who want to implement the library rather than to use it.

Note that the language has changed during the standardization of C++11, just as it changed during the standardization of C++98, so your knowledge might not be up-to-date. Chapter 3 provides a brief overview of and introduction to the latest language features that are important for using the C++11 library. Many of the new library features use these new language features, so you should read Chapter 3 to review all the new features of C++. But I will also refer to that chapter when libraries use new language features.

1.3 Style and Structure of the Book

The C++ standard library provides components that are somewhat, but not totally, independent of one another, so there is no easy way to describe each part without mentioning others. I considered various approaches for presenting the contents of this book. One was on the order of the C++ standard. However, this is not the best way to explain the components of the C++ standard library from scratch. Another approach was to start with an overview of all components, followed by chapters that provided more details. Alternatively, I could have sorted the components, trying to find an order that had a minimum of cross-references to other sections. My solution was to use a mixture of all three approaches. I start with a brief introduction of the general concepts and the utilities that the library uses. Then, I describe all the components, each in one or more chapters. The first component is the standard template library (STL). There is no doubt that the STL is the most powerful, most complex, and most exciting part of the library. Its design influences other components heavily. Then, I describe the more self-explanatory components, such as special containers, strings, and regular expressions. The next component discussed is one you probably know and use already: the `IOStream` library. That component is followed by a discussion of internationalization, which had some influence on the `IOStream` library. Finally, I describe the library parts dealing with numerics, concurrency, and allocators.

Each component description begins with the component's purpose, design, and some examples. Next, a detailed description begins with various ways to use the component, as well as any traps and pitfalls associated with it. The description usually ends with a reference section, in which you can find the exact signature and definition of a component's classes and its functions.

List of Contents

The first five chapters introduce this book and the C++ standard library in general:

- **Chapter 1: About This Book** introduces the book’s subject and describes its contents.
- **Chapter 2: Introduction to C++ and the Standard Library** provides a brief overview of the history of the C++ standard library and the context of its standardization and introduces the concept of complexity.
- **Chapter 3: New Language Features** provides an overview of the new language features you should know to read this book and to use the C++ standard library.
- **Chapter 4: General Concepts** describes the fundamental library concepts that you need to understand to work with all the components. In particular, the chapter introduces the namespace `std`, the format of header files, and the general support of error and exception handling.
- **Chapter 5: Utilities** describes several small utilities provided for the user of the library and for the library itself. In particular, the chapter describes classes `pair<>` and `tuple<>`, smart pointers, numeric limits, type traits and type utilities, auxiliary functions, class `ratio<>`, clocks and timers, and available C functions.

Chapters 6 through 11 describe all aspects of the STL:

- **Chapter 6: The Standard Template Library** presents a detailed introduction to the concept of the STL, which provides container classes and algorithms that are used to process collections of data. The chapter explains step-by-step the concept, the problems, and the special programming techniques of the STL, as well as the roles of its parts.
- **Chapter 7: STL Containers** explains the concepts and describes the abilities of the STL’s container classes. The chapter describes arrays, vectors, deques, lists, forward lists, sets, maps, and unordered containers with their common abilities, differences, specific benefits, and drawbacks and provides typical examples.
- **Chapter 8: STL Container Members in Detail** lists and describes all container members (types and operations) in the form of a handy reference.
- **Chapter 9: STL Iterators** explains the various iterator categories, the auxiliary functions for iterators, and the iterator adapters, such as stream iterators, reverse iterators, insert iterators, and move iterators.
- **Chapter 10: STL Function Objects and Using Lambdas** details the STL’s function object classes, including lambdas, and how to use them to define the behavior of containers and algorithms.
- **Chapter 11: STL Algorithms** lists and describes the STL’s algorithms. After a brief introduction and comparison of the algorithms, each algorithm is described in detail, followed by one or more example programs.

Chapters 12 through 14 describe “simple” individual standard classes of the C++ standard library:

- **Chapter 12: Special Containers** describes the container adapters for queues and stacks, as well as the class `bitset`, which manages a bitfield with an arbitrary number of bits or flags.
- **Chapter 13: Strings** describes the string types of the C++ standard library (yes, there are more than one). The standard provides strings as “kind of” fundamental data types with the ability to use different types of characters.

- **Chapter 14: Regular Expressions** describes the interface to deal with regular expressions, which can be used to search and replace characters and substrings.

Chapters 15 and 16 deal with the two closely related subjects of I/O and internationalization:

- **Chapter 15: Input/Output Using Stream Classes** covers the standardized form of the commonly known `IOStream` library. The chapter also describes details that are typically not so well known but that may be important to programmers, such as the correct way to define and integrate special I/O channels.
- **Chapter 16: Internationalization** covers the concepts and classes for the internationalization of programs, such as the handling of different character sets and the use of different formats for floating-point numbers and dates.

The remaining chapters cover numerics, concurrency, and allocators:

- **Chapter 17: Numerics** describes the numeric components of the C++ standard library: in particular, classes for random numbers and distributions, types for complex numbers, and some numeric C functions.
- **Chapter 18: Concurrency** describes the features provided by the C++ standard library to enable and support concurrency and multithreading.
- **Chapter 19: Allocators** describes the concept of different memory models in the C++ standard library.

The book concludes with a **bibliography** and an **index**.

Due to the size of this book I had to move material that is not so relevant for the average application programmer but should be covered to a **supplementary chapter** provided on the Web site of this book: <http://www.cppstdlib.com>. That material includes:

- Details of bitsets (introduced in Section 12.5)
- Class `valarray<>` (very briefly introduced in Section 17.4)
- Details of allocators (introduced in Chapter 19)

1.4 How to Read This Book

This book is both an introductory user's guide and a structured reference manual about the C++ standard library. The individual components of the C++ standard library are somewhat independent of one another, so after reading Chapters 2 through 5 you could read the chapters that discuss the individual components in any order. Bear in mind that Chapters 6 through 11 all describe the same component. To understand the other STL chapters, you should start with the introduction to the STL in Chapter 6.

If you are a C++ programmer who wants to know, in general, the concepts and all parts of the library, you could simply read the book from beginning to end. However, you should skip the reference sections. To program with certain components of the C++ standard library, the best way to find something is to use the index, which I have tried to make comprehensive enough to save you time when you are looking for something.

In my experience, the best way to learn something new is to look at examples. Therefore, you'll find a lot of examples throughout the book. They may be a few lines of code or complete programs.

In the latter case, you'll find the name of the file containing the program as the first comment line. You can find the files on the Internet at the Web site of the book: <http://www.cppstdlib.com>.

1.5 State of the Art

The C++11 standard was completed while I was writing this book. Please bear in mind that some compilers might not yet conform to the standard. This will most likely change in the near future. As a consequence, you might discover that not all things covered in this book work as described on your system, and you may have to change example programs to fit your specific environment.

1.6 Example Code and Additional Information

You can access all example programs and acquire more information about this book and the C++ standard library from my Web site: <http://www.cppstdlib.com>. Also, you can find a lot of additional information about this topic on the Internet. See the bibliography, which is also provided on the Web site, for some of them.

1.7 Feedback

I welcome your feedback (good and bad) on this book. I tried to prepare it carefully; however, I'm human, and at some point I have to stop writing and tweaking. So, you may find some errors, inconsistencies, or subjects that could be described better. Your feedback will give me the chance to improve later editions.

The best way to reach me is by email. However, to avoid spam problems, I haven't included an email address inside this book. (I had to stop using the email address I put in the first edition after I started getting thousands of spam emails per day.) Please refer to the book's Web site, <http://www.cppstdlib.com>, to get an email address for feedback.

Many thanks.