

Contents

Preface	xvii
Acknowledgments	xix
1 About this Book	1
1.1 Why this Book	1
1.2 What You Should Know 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	7
2.2 New Language Features	9
2.2.1 Templates	9
2.2.2 Explicit Initialization for Fundamental Types	14
2.2.3 Exception Handling	15
2.2.4 Namespaces	16
2.2.5 Type <code>bool</code>	18
2.2.6 Keyword <code>explicit</code>	18
2.2.7 New Operators for Type Conversion	19
2.2.8 Initialization of Constant Static Members	20
2.2.9 Definition of <code>main()</code>	21
2.3 Complexity and the Big-O Notation	21

3	General Concepts	23
3.1	Namespace <code>std</code>	23
3.2	Header Files	24
3.3	Error and Exception Handling	25
3.3.1	Standard Exception Classes	25
3.3.2	Members of Exception Classes	28
3.3.3	Throwing Standard Exceptions	29
3.3.4	Deriving Standard Exception Classes	30
3.4	Allocators	31
4	Utilities	33
4.1	Pairs	33
4.1.1	Convenience Function <code>make_pair()</code>	36
4.1.2	Examples of Pair Usage	37
4.2	Class <code>auto_ptr</code>	38
4.2.1	Motivation of Class <code>auto_ptr</code>	38
4.2.2	Transfer of Ownership by <code>auto_ptr</code>	40
4.2.3	<code>auto_ptr</code> s as Members	44
4.2.4	Misusing <code>auto_ptr</code> s	46
4.2.5	<code>auto_ptr</code> Examples	47
4.2.6	Class <code>auto_ptr</code> in Detail	51
4.3	Numeric Limits	59
4.4	Auxiliary Functions	66
4.4.1	Processing the Minimum and Maximum	66
4.4.2	Swapping Two Values	67
4.5	Supplementary Comparison Operators	69
4.6	Header Files <code><cstdlib></code> and <code><cstdliblib></code>	71
4.6.1	Definitions in <code><cstdlib></code>	71
4.6.2	Definitions in <code><cstdliblib></code>	71
5	The Standard Template Library	73
5.1	STL Components	73
5.2	Containers	75
5.2.1	Sequence Containers	76
5.2.2	Associative Containers	81
5.2.3	Container Adapters	82

5.3	Iterators	83
5.3.1	Examples of Using Associative Containers	86
5.3.2	Iterator Categories	93
5.4	Algorithms	94
5.4.1	Ranges	97
5.4.2	Handling Multiple Ranges	101
5.5	Iterator Adapters	104
5.5.1	Insert Iterators	104
5.5.2	Stream Iterators	107
5.5.3	Reverse Iterators	109
5.6	Manipulating Algorithms	111
5.6.1	“Removing” Elements	111
5.6.2	Manipulating Algorithms and Associative Containers	115
5.6.3	Algorithms versus Member Functions	116
5.7	User-Defined Generic Functions	117
5.8	Functions as Algorithm Arguments	119
5.8.1	Examples of Using Functions as Algorithm Arguments	119
5.8.2	Predicates	121
5.9	Function Objects	124
5.9.1	What Are Function Objects?	124
5.9.2	Predefined Function Objects	131
5.10	Container Elements	134
5.10.1	Requirements for Container Elements	134
5.10.2	Value Semantics or Reference Semantics	135
5.11	Errors and Exceptions Inside the STL	136
5.11.1	Error Handling	137
5.11.2	Exception Handling	139
5.12	Extending the STL	141
6	STL Containers	143
6.1	Common Container Abilities and Operations	144
6.1.1	Common Container Abilities	144
6.1.2	Common Container Operations	144
6.2	Vectors	148
6.2.1	Abilities of Vectors	148
6.2.2	Vector Operations	150

6.2.3	Using Vectors as Ordinary Arrays	155
6.2.4	Exception Handling	155
6.2.5	Examples of Using Vectors	156
6.2.6	Class <code>vector<bool></code>	158
6.3	Deque	160
6.3.1	Abilities of Deques	161
6.3.2	Deque Operations	162
6.3.3	Exception Handling	164
6.3.4	Examples of Using Deques	164
6.4	Lists	166
6.4.1	Abilities of Lists	166
6.4.2	List Operations	167
6.4.3	Exception Handling	172
6.4.4	Examples of Using Lists	172
6.5	Sets and Multisets	175
6.5.1	Abilities of Sets and Multisets	176
6.5.2	Set and Multiset Operations	177
6.5.3	Exception Handling	185
6.5.4	Examples of Using Sets and Multisets	186
6.5.5	Example of Specifying the Sorting Criterion at Runtime	191
6.6	Maps and Multimaps	194
6.6.1	Abilities of Maps and Multimaps	195
6.6.2	Map and Multimap Operations	196
6.6.3	Using Maps as Associative Arrays	205
6.6.4	Exception Handling	207
6.6.5	Examples of Using Maps and Multimaps	207
6.6.6	Example with Maps, Strings, and Sorting Criterion at Runtime	213
6.7	Other STL Containers	217
6.7.1	Strings as STL Containers	217
6.7.2	Ordinary Arrays as STL Containers	218
6.7.3	Hash Tables	221
6.8	Implementing Reference Semantics	222
6.9	When to Use which Container	226
6.10	Container Types and Members in Detail	230
6.10.1	Type Definitions	230
6.10.2	Create, Copy, and Destroy Operations	231

6.10.3	Nonmodifying Operations	233
6.10.4	Assignments	236
6.10.5	Direct Element Access	237
6.10.6	Operations to Generate Iterators	239
6.10.7	Inserting and Removing Elements	240
6.10.8	Special Member Functions for Lists	244
6.10.9	Allocator Support	246
6.10.10	Overview of Exception Handling in STL Containers	248
7	STL Iterators	251
7.1	Header Files for Iterators	251
7.2	Iterator Categories	251
7.2.1	Input Iterators	252
7.2.2	Output Iterators	253
7.2.3	Forward Iterators	254
7.2.4	Bidirectional Iterators	255
7.2.5	Random Access Iterators	255
7.2.6	The Increment and Decrement Problem of Vector Iterators	258
7.3	Auxiliary Iterator Functions	259
7.3.1	Stepping Iterators Using <code>advance()</code>	259
7.3.2	Processing Iterator Distance Using <code>distance()</code>	261
7.3.3	Swapping Iterator Values Using <code>iter_swap()</code>	263
7.4	Iterator Adapters	264
7.4.1	Reverse Iterators	264
7.4.2	Insert Iterators	271
7.4.3	Stream Iterators	277
7.5	Iterator Traits	283
7.5.1	Writing Generic Functions for Iterators	285
7.5.2	User-Defined Iterators	288
8	STL Function Objects	293
8.1	The Concept of Function Objects	293
8.1.1	Function Objects as Sorting Criteria	294
8.1.2	Function Objects with Internal State	296
8.1.3	The Return Value of <code>for_each()</code>	300
8.1.4	Predicates versus Function Objects	302

8.2	Predefined Function Objects	305
8.2.1	Function Adapters	306
8.2.2	Function Adapters for Member Functions	307
8.2.3	Function Adapters for Ordinary Functions	309
8.2.4	User-Defined Function Objects for Function Adapters	310
8.3	Supplementary Composing Function Objects	313
8.3.1	Unary Compose Function Object Adapters	314
8.3.2	Binary Compose Function Object Adapters	318
9	STL Algorithms	321
9.1	Algorithm Header Files	321
9.2	Algorithm Overview	322
9.2.1	A Brief Introduction	322
9.2.2	Classification of Algorithms	323
9.3	Auxiliary Functions	332
9.4	The <code>for_each()</code> Algorithm	334
9.5	Nonmodifying Algorithms	338
9.5.1	Counting Elements	338
9.5.2	Minimum and Maximum	339
9.5.3	Searching Elements	341
9.5.4	Comparing Ranges	356
9.6	Modifying Algorithms	363
9.6.1	Copying Elements	363
9.6.2	Transforming and Combining Elements	366
9.6.3	Swapping Elements	370
9.6.4	Assigning New Values	372
9.6.5	Replacing Elements	375
9.7	Removing Algorithms	378
9.7.1	Removing Certain Values	378
9.7.2	Removing Duplicates	381
9.8	Mutating Algorithms	386
9.8.1	Reversing the Order of Elements	386
9.8.2	Rotating Elements	388
9.8.3	Permuting Elements	391
9.8.4	Shuffling Elements	393
9.8.5	Moving Elements to the Front	395

9.9	Sorting Algorithms	397
9.9.1	Sorting All Elements	397
9.9.2	Partial Sorting	400
9.9.3	Sorting According to the n th Element	404
9.9.4	Heap Algorithms	406
9.10	Sorted Range Algorithms	409
9.10.1	Searching Elements	410
9.10.2	Merging Elements	416
9.11	Numeric Algorithms	425
9.11.1	Processing Results	425
9.11.2	Converting Relative and Absolute Values	429
10	Special Containers	435
10.1	Stacks	435
10.1.1	The Core Interface	436
10.1.2	Example of Using Stacks	437
10.1.3	Class <code>stack<></code> in Detail	438
10.1.4	A User-Defined Stack Class	441
10.2	Queues	444
10.2.1	The Core Interface	445
10.2.2	Example of Using Queues	446
10.2.3	Class <code>queue<></code> in Detail	447
10.2.4	A User-Defined Queue Class	450
10.3	Priority Queues	453
10.3.1	The Core Interface	455
10.3.2	Example of Using Priority Queues	455
10.3.3	Class <code>priority_queue<></code> in Detail	456
10.4	Bitsets	460
10.4.1	Examples of Using Bitsets	460
10.4.2	Class <code>bitset</code> in Detail	463
11	Strings	471
11.1	Motivation	471
11.1.1	A First Example: Extracting a Temporary File Name	472
11.1.2	A Second Example: Extracting Words and Printing Them Backward	476
11.2	Description of the String Classes	479
11.2.1	String Types	479

11.2.2	Operation Overview	481
11.2.3	Constructors and Destructors	483
11.2.4	Strings and C-Strings	484
11.2.5	Size and Capacity	485
11.2.6	Element Access	487
11.2.7	Comparisons	488
11.2.8	Modifiers	489
11.2.9	Substrings and String Concatenation	492
11.2.10	Input/Output Operators	492
11.2.11	Searching and Finding	493
11.2.12	The Value <code>npos</code>	495
11.2.13	Iterator Support for Strings	497
11.2.14	Internationalization	503
11.2.15	Performance	506
11.2.16	Strings and Vectors	506
11.3	String Class in Detail	507
11.3.1	Type Definitions and Static Values	507
11.3.2	Create, Copy, and Destroy Operations	508
11.3.3	Operations for Size and Capacity	510
11.3.4	Comparisons	511
11.3.5	Character Access	512
11.3.6	Generating C-Strings and Character Arrays	513
11.3.7	Modifying Operations	514
11.3.8	Searching and Finding	520
11.3.9	Substrings and String Concatenation	524
11.3.10	Input/Output Functions	524
11.3.11	Generating Iterators	525
11.3.12	Allocator Support	526
12	Numerics	529
12.1	Complex Numbers	529
12.1.1	Examples Using Class <code>Complex</code>	530
12.1.2	Operations for Complex Numbers	533
12.1.3	Class <code>complex<></code> in Detail	541
12.2	Valarrays	547
12.2.1	Getting to Know Valarrays	547
12.2.2	Valarray Subsets	553

12.2.3	Class <code>valarray</code> in Detail	569
12.2.4	<code>Valarray</code> Subset Classes in Detail	575
12.3	Global Numeric Functions	581
13	Input/Output Using Stream Classes	583
13.1	Common Background of I/O Streams	584
13.1.1	Stream Objects	584
13.1.2	Stream Classes	584
13.1.3	Global Stream Objects	585
13.1.4	Stream Operators	586
13.1.5	Manipulators	586
13.1.6	A Simple Example	587
13.2	Fundamental Stream Classes and Objects	588
13.2.1	Classes and Class Hierarchy	588
13.2.2	Global Stream Objects	591
13.2.3	Header Files	592
13.3	Standard Stream Operators <code><<</code> and <code>>></code>	593
13.3.1	Output Operator <code><<</code>	593
13.3.2	Input Operator <code>>></code>	594
13.3.3	Input/Output of Special Types	595
13.4	State of Streams	597
13.4.1	Constants for the State of Streams	597
13.4.2	Member Functions Accessing the State of Streams	598
13.4.3	Stream State and Boolean Conditions	600
13.4.4	Stream State and Exceptions	602
13.5	Standard Input/Output Functions	607
13.5.1	Member Functions for Input	607
13.5.2	Member Functions for Output	610
13.5.3	Example Uses	611
13.6	Manipulators	612
13.6.1	How Manipulators Work	612
13.6.2	User-Defined Manipulators	614
13.7	Formatting	615
13.7.1	Format Flags	615
13.7.2	Input/Output Format of Boolean Values	617
13.7.3	Field Width, Fill Character, and Adjustment	618

13.7.4	Positive Sign and Uppercase Letters	620
13.7.5	Numeric Base	621
13.7.6	Floating-Point Notation	623
13.7.7	General Formatting Definitions	625
13.8	Internationalization	625
13.9	File Access	627
13.9.1	File Flags	631
13.9.2	Random Access	634
13.9.3	Using File Descriptors	637
13.10	Connecting Input and Output Streams	637
13.10.1	Loose Coupling Using <code>tie()</code>	637
13.10.2	Tight Coupling Using Stream Buffers	638
13.10.3	Redirecting Standard Streams	641
13.10.4	Streams for Reading and Writing	643
13.11	Stream Classes for Strings	645
13.11.1	String Stream Classes	645
13.11.2	<code>char*</code> Stream Classes	649
13.12	Input/Output Operators for User-Defined Types	652
13.12.1	Implementing Output Operators	652
13.12.2	Implementing Input Operators	654
13.12.3	Input/Output Using Auxiliary Functions	656
13.12.4	User-Defined Operators Using Unformatted Functions	658
13.12.5	User-Defined Format Flags	659
13.12.6	Conventions for User-Defined Input/Output Operators	662
13.13	The Stream Buffer Classes	663
13.13.1	User's View of Stream Buffers	663
13.13.2	Stream Buffer Iterators	665
13.13.3	User-Defined Stream Buffers	668
13.14	Performance Issues	681
13.14.1	Synchronization with C's Standard Streams	682
13.14.2	Buffering in Stream Buffers	682
13.14.3	Using Stream Buffers Directly	683
14	Internationalization	685
14.1	Different Character Encodings	686
14.1.1	Wide-Character and Multibyte Text	686

14.1.2	Character Traits	687
14.1.3	Internationalization of Special Characters	691
14.2	The Concept of Locales	692
14.2.1	Using Locales	693
14.2.2	Locale Facets	698
14.3	Locales in Detail	700
14.4	Facets in Detail	704
14.4.1	Numeric Formatting	705
14.4.2	Time and Date Formatting	708
14.4.3	Monetary Formatting	711
14.4.4	Character Classification and Conversion	715
14.4.5	String Collation	724
14.4.6	Internationalized Messages	725
15	Allocators	727
15.1	Using Allocators as an Application Programmer	727
15.2	Using Allocators as a Library Programmer	728
15.3	The Default Allocator	732
15.4	A User-Defined Allocator	735
15.5	Allocators in Detail	737
15.5.1	Type Definitions	737
15.5.2	Operations	739
15.6	Utilities for Uninitialized Memory in Detail	740
	Internet Resources	743
	Bibliography	745
	Index	747

Preface

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 — an English book 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 every-day 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

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 Meyers, 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 Rumbsby, 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.std.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.

The standardization process was started in 1989 by an international ANSI/ISO committee. It developed the standard based on Bjarne Stroustrup's books *The C++ Programming Language* and *The Annotated C++ Reference Manual*. After the standard was completed in 1997, several formal motions by different countries made it an international ISO and ANSI standard in 1998. The standardization process included the development of a C++ standard library. The library extends the core language to provide some general components. By using C++'s ability to program new abstract and generic types, the library provides a set of common classes and interfaces. This gives programmers a higher level of abstraction. The library provides the ability to use

- String types
- Different data structures (such as dynamic arrays, linked lists, and binary trees)
- Different algorithms (such as different sorting algorithms)
- Numeric classes
- Input/output (I/O) classes
- Classes for internationalization support

All of these are supported by a fairly simple programming interface. These components are very important for many programs. These days, data processing often means inputting, computing, processing, and outputting large amounts of data, which are often strings.

The library is not self-explanatory. To use these components and to benefit from their power, you need a good 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 of its components from a conceptual point of view. Next, it describes the details needed for practical programming. Examples are included to demonstrate the exact usage of the components. Thus, this book is a detailed introduction to the C++ library for both the beginner and the practical 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. Beware that the library has traps and pitfalls, which I point out when we encounter them and suggest ways of avoiding them.

1.2 What You Should Know 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, and exception handling. However, you don't have to know all of 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 use it). Note that the language has changed during the standardization process, so your knowledge might not be up to date. Section 2.2, page 9, provides a brief overview and introduction of the latest language features that are important for using the library. You should read this section if you are not sure whether you know all the new features of C++ (such as the keyword `typename` and the concept of namespaces).

1.3 Style and Structure of the Book

The C++ standard library provides different components that are somewhat but not totally independent of each other, so there is no easy way to describe each part without mentioning others. I considered several different 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 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 are used by the library. 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 numeric classes. The next component discussed is one you probably know and use already: the `IOStream` library. It is followed by a discussion of internationalization, which had some influence on the `IOStream` library.

Each component description begins with the component's purpose, design, and some examples. Next, a detailed description follows that begins with different 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.

The following is a description of the book's contents. The first four chapters introduce this book and the C++ standard library in general:

- **Chapter 1: About this Book**

This chapter (which you are reading right now) introduces the book's subject and describes its contents.

- **Chapter 2: Introduction to C++ and the Standard Library**

This chapter provides a brief overview of the history of the C++ standard library and the context of its standardization. It also contains some general hints regarding the technical background for this book and the library, such as new language features and the concept of complexity.

- **Chapter 3: General Concepts**

This chapter describes the fundamental concepts of the library that you need to understand to work with all the components. In particular, it introduces the namespace `std`, the format of header files, and the general support of error and exception handling.

- **Chapter 4: Utilities**

This chapter describes several small utilities provided for the user of the library and for the library itself. In particular, it describes auxiliary functions such as `max()`, `min()`, and `swap()`, types `pair` and `auto_ptr`, as well as `numeric_limits`, which provide more information about implementation-specific details of numeric data types.

Chapters 5 through 9 describe all aspects of the STL:

- **Chapter 5: The Standard Template Library**

This chapter presents a detailed introduction to the concept of the STL, which provides container classes and algorithms that are used to process collections of data. It 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 6: STL Containers**

This chapter explains the concepts and describes the abilities of the STL's container classes. First it describes the differences between vectors, deques, lists, sets, and maps, then their common abilities, and all with typical examples. Lastly it lists and describes all container functions in form of a handy reference.

- **Chapter 7: STL Iterators**

This chapter deals in detail with the STL's iterator classes. In particular, it explains the different iterator categories, the auxiliary functions for iterators, and the iterator adapters, such as stream iterators, reverse iterators, and insert iterators.

- **Chapter 8: STL Function Objects**

This chapter details the STL's function object classes.

- **Chapter 9: STL Algorithms**

This chapter 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 10 through 12 describe “simple” individual standard classes:

- **Chapter 10: Special Containers**

This chapter describes the different special container classes of the C++ standard library. It covers 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 11: Strings**

This chapter describes the string types of the C++ standard library (yes, there are more than one). The standard provides strings as kind of “self-explanatory” fundamental data types with the ability to use different types of characters.

- **Chapter 12: Numerics**

This chapter describes the numeric components of the C++ standard library. In particular, it covers types for complex numbers and classes for the processing of arrays of numeric values (the latter may be used for matrices, vectors, and equations).

Chapters 13 and 14 deal with I/O and internationalization (two closely related subjects):

- **Chapter 13: Input/Output Using Stream Classes**

This chapter covers the I/O component of C++. This component is the standardized form of the commonly known `IOStream` library. The chapter also describes details that may be important to programmers but are typically not so well known. For example, it describes the correct way to define and integrate special I/O channels, which are often implemented incorrectly in practice.

- **Chapter 14: Internationalization**

This chapter covers the concepts and classes for the internationalization of programs. In particular, it describes the handling of different character sets, as well as the use of different formats for such values as floating-point numbers and dates.

The rest of the book contains:

- **Chapter 15: Allocators**

This chapter describes the concept of different memory models in the C++ standard library.

- An **appendix** with

- **Internet Resources**
- **Bibliography**
- **Index**

1.4 How to Read this Book

This book is a mix of introductory user’s guide and structured reference manual regarding the C++ standard library. The individual components of the C++ standard library are independent of each other, to some extent, so after reading Chapters 2 through 4 you could read the chapters that discuss the individual components in any order. Bear in mind, that Chapter 5 through Chapter 9 all describe the same component. To understand the other STL chapters, you should start with the introduction to the STL in Chapter 5.

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 the beginning to the 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. I have tried to make the index very comprehensive 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 my Web site at <http://www.josuttis.com/libbook/>.

1.5 State of the Art

While I was writing this book, the C++ standard was completed. Please bear in mind that some compilers might not yet confirm to it. 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. I can compile almost all example programs with version 2.8 or higher of the EGCS compiler, which is free for almost all platforms and available on the Internet (see <http://egcs.cygnum.com/>) and on several software CDs.

1.6 Example Code and Additional Information

You can access all example programs and acquire more informations about this book and the C++ standard library from my Web site at <http://www.josuttis.com/libbook/>. Also, you can find a lot of additional information about this topic on the Internet. See Internet Resources on page 743 for details.

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 time 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:

`libbook@josuttis.com`

You can also reach me by phone, fax, or “snail” mail:

Nicolai M. Josuttis
Berggarten 9
D-38108 Braunschweig
Germany
Phone: +49 5309 5747
Fax: +49 5309 5774

Many thanks.