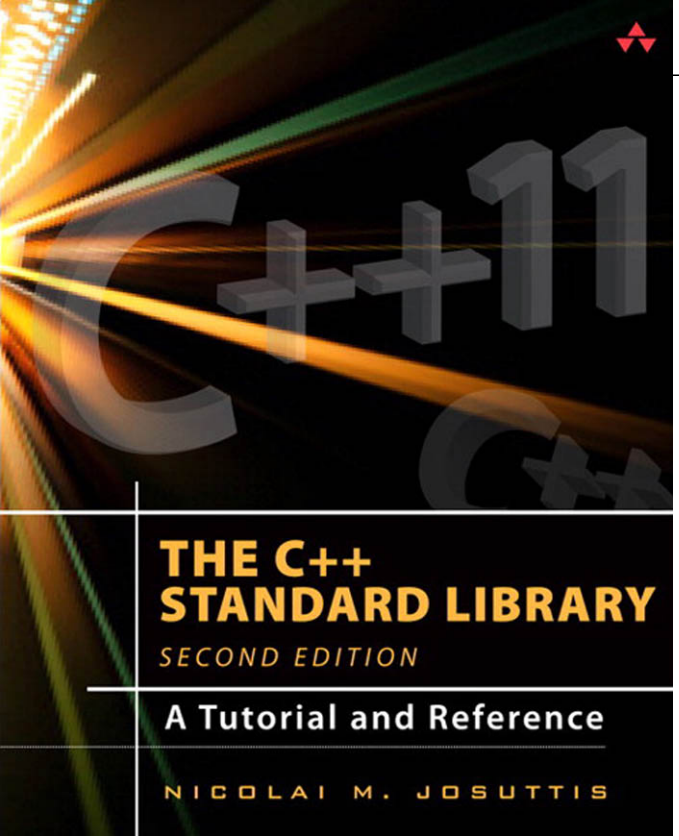
C++11
THE C++
STANDARD LIBRARY
SECOND EDITION
A Tutorial and Reference
NICOLAI M. JOSUTTIS

The C++ Standard Library

Second Edition

This page intentionally left blank

The C++ Standard Library

A Tutorial and Reference

Second Edition

Nicolai M. Josuttis

Addison-Wesley

Upper Saddle River, NJ • Boston • Indianapolis • San Francisco
New York • Toronto • Montreal • London • Munich • Paris • Madrid
Capetown • Sydney • Tokyo • Singapore • Mexico City

Many of the designations used by manufacturers and sellers to distinguish their products are claimed as trademarks. Where those designations appear in this book, and the publisher was aware of a trademark claim, the designations have been printed with initial capital letters or in all capitals.

The author and publisher have taken care in the preparation of this book, but make no expressed or implied warranty of any kind and assume no responsibility for errors or omissions. No liability is assumed for incidental or consequential damages in connection with or arising out of the use of the information or programs contained herein.

The publisher offers excellent discounts on this book when ordered in quantity for bulk purchases or special sales, which may include electronic versions and/or custom covers and content particular to your business, training goals, marketing focus, and branding interests. For more information, please contact:

U.S. Corporate and Government Sales
(800) 382-3419
corpsales@pearsontechgroup.com

For sales outside the United States, please contact:

International Sales
international@pearson.com

Visit us on the Web: informit.com/aw

Library of Congress Cataloging-in-Publication Data

Josuttis, Nicolai M.
The C++ standard library : a tutorial and reference / Nicolai M. Josuttis.—2nd ed.
p. cm.
Includes bibliographical references and index.
ISBN 978-0-321-62321-8 (hardcover : alk. paper)
1. C++ (Computer program language) I. Title.
QA76.73.C153J69 2012
005.13'3-dc23

2011045071

Copyright © 2012 Pearson Education, Inc.

This book was typeset by the author using the LaTeX document processing system.

All rights reserved. Printed in the United States of America. This publication is protected by copyright, and permission must be obtained from the publisher prior to any prohibited reproduction, storage in a retrieval system, or transmission in any form or by any means, electronic, mechanical, photocopying, recording, or likewise. To obtain permission to use material from this work, please submit a written request to Pearson Education, Inc., Permissions Department, One Lake Street, Upper Saddle River, New Jersey 07458, or you may fax your request to (201) 236-3290.

ISBN-13: 978-0-321-62321-8
ISBN-10: 0-321-62321-5

Text printed in the United States on recycled paper at Edwards Brothers in Ann Arbor, Michigan.
First printing, March 2012

To those who care
for people and mankind

This page intentionally left blank

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 `auto` 14
3.1.3 Uniform Initialization and Initializer Lists 15
3.1.4 Range-Based `for` Loops 17
3.1.5 Move Semantics and Rvalue References 19

3.1.6 New String Literals . . . 23
3.1.7 Keyword `noexcept` . . . 24
3.1.8 Keyword `constexpr` . . . 26
3.1.9 New Template Features . . . 26
3.1.10 Lambdas . . . 28
3.1.11 Keyword `decltype` . . . 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 `main()` . . . 37

4 General Concepts **39**
4.1 Namespace `std` . . . 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 `exception_ptr` . . . 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 `tuples` and `pairs` . . . 75
5.2 Smart Pointers . . . 76
5.2.1 Class `shared_ptr` . . . 76
5.2.2 Class `weak_ptr` . . . 84
5.2.3 Misusing Shared Pointers . . . 89
5.2.4 Shared and Weak Pointers in Detail . . . 92
5.2.5 Class `unique_ptr` . . . 98

5.2.6 Class unique_ptr in Detail . . . 110
5.2.7 Class auto_ptr . . . 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 ratio<> . . . 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 <cstddef>, <cstdlib>, and <cstring> . . . 161
5.8.1 Definitions in <cstddef> . . . 161
5.8.2 Definitions in <cstdlib> . . . 162
5.8.3 Definitions in <cstring> . . . 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 `array`s 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 `vector<bool>` . . . 281
7.4 Deques . . . 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 `advance()` . . . 441
9.3.2 `next()` and `prev()` . . . 443
9.3.3 `distance()` . . . 445
9.3.4 `iter_swap()` . . . 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 `for_each()` . . . 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 `for_each()` 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 `stack<>` 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 `queue<>` 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 `priority_queue<>` 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 `bitset` 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 `npos` . . . 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 `tie()` . . . 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 `complex<>` in General . . . 925
17.2.2 Examples Using Class `complex<>` . . . 926
17.2.3 Operations for Complex Numbers . . . 928
17.2.4 Class `complex<>` in Detail . . . 935
17.3 Global Numeric Functions . . . 941
17.4 Valarrays . . . 943

18 Concurrency **945**
18.1 The High-Level Interface: `async()` and Futures . . . 946
18.1.1 A First Example Using `async()` 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 `std::thread` . . . 964
18.2.2 Promises . . . 969
18.2.3 Class `packaged_task<>` . . . 972
18.3 Starting a Thread in Detail . . . 973
18.3.1 `async()` in Detail . . . 974
18.3.2 Futures in Detail . . . 975
18.3.3 Shared Futures in Detail . . . 976
18.3.4 Class `std::promise` in Detail . . . 977
18.3.5 Class `std::packaged_task` in Detail . . . 977
18.3.6 Class `std::thread` in Detail . . . 979
18.3.7 Namespace `this_thread` . . . 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**

This page intentionally left blank

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.

sample content of The C++ Standard Library: A Tutorial and Reference (2nd Edition)

- Why Didn't They Ask Evans? for free
- read Obsessed (Peter Zak, Book 4)
- **Eat, Read, Love: A Cookbook book**
- **read The Warrior King (The Seer King Trilogy, Book 3) online**
- Guerrilla Marketing For Dummies book
- CCNA Security Lab Manual for free

- http://jaythebody.com/freebooks/The-Ghost-and-the-Document-Reviewer--A-Mystery-Novella.pdf
- http://test.markblaustein.com/library/Obsessed--Peter-Zak--Book-4-.pdf
- http://patrickvincitore.com/?ebooks/Children-Playing-Before-a-Statue-of-Hercules.pdf
- http://nexson.arzamaszev.com/library/Beasts-and-Super-Beasts.pdf
- http://nexson.arzamaszev.com/library/Guerrilla-Marketing-For-Dummies.pdf
- http://paulczajak.com/?library/Black-Horse-Creek.pdf