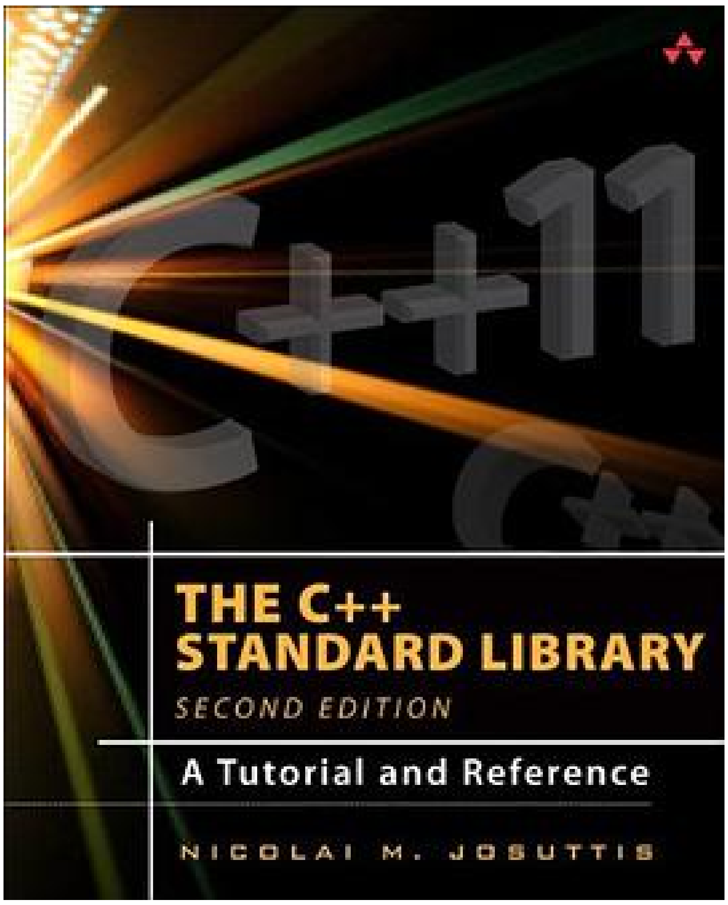
The C++ Standard Library, 2nd Edition



[The C++ Standard Library, 2nd Edition_下载链接1_]

著者:[德] Nicolai M・Josuttis

出版者:Addison-Wesley Professional

出版时间:2012-4-9

装帧:Hardcover

isbn:9780321623218

The Best-Selling Programmer Resource–Now Updated for C++11

The C++ standard library provides a set of common classes and interfaces that greatly extend the core C++ language. The library, however, is not self-explanatory. To make full use of its components - and to benefit from their power - you need a resource that does far more than list the classes and their functions.

The C++ Standard Library - A Tutorial and Reference, 2nd Edition describes this library as now incorporated into the new ANSI/ISO C++ language standard (C++11). The book provides comprehensive documentation of each library component, including an introduction to its purpose and design; clearly written explanations of complex concepts; the practical programming details needed for effective use; traps and pitfalls; the exact signature and definition of the most important classes and functions; and numerous examples of working code.

The book focuses on the Standard Template Library (STL), examining containers, iterators, function objects, and STL algorithms. You will also find detailed coverage of strings, concurrency, random numbers and distributions, special containers, numerical classes, internationalization, and the IOStreams library. An insightful introduction to fundamental concepts and an overview of the library will help bring newcomers quickly up to speed. A comprehensive index will support the C++ programmer in his/her day-to-day life.

The book covers all the new C++11 library components, including

Concurrency

Fractional arithmetic

Clocks and Timers

Random numbers and distributions

New smart pointers

Regular expressions

New STL containers, such as arrays, forward lists, and unordered containers

New STL algorithms

Tuples

Type traits and type utilities

The book also examines the new C++ programming style and its effect on the standard library, including lambdas, range-based for loops, and variadic templates.

An accompanying Web site, including source code, can be found at http://www.josuttis.com/.

作者介绍:

Nicolai M. Josuttis is an independent technical consultant who designs mid-sized and large software systems for the telecommunication, traffic, finance, and manufacturing industries. A former member of the C++ Standard Committee library working group, Nico is well known in the programming community for his authoritative books. In addition to The C++ Standard Library, a worldwide best-seller since its first publication in 1999, his books include C++ Templates: The Complete Guide (with David Vandevoorde, Addison-Wesley, 2003) and SOA in Practice: The Art of Distributed System Design (O’Reilly Media, 2007)

目录: Preface
Acknowledgments
--
1 About this Book
1.1 Why this Book
1.2 What You Should Know Before Reading this Book
1.3 Style and Structure of the Book
1.4 How to Read this Book
1.5 State of the Art
1.6 Example Code and Additional Information
1.7 Feedback
--
2 Introduction to C++ and the Standard Library
2.1 History
2.2 New Language Features
2.2.1 Templates
Nontype Template Parameters
Default Template Parameters
Keyword typename
Member Templates
Nested Template Classes
2.2.2 Explicit Initialization for Fundamental Types
2.2.3 Exception Handling
2.2.4 Namespaces
2.2.5 Type bool
2.2.6 Keyword explicit
2.2.7 New Operators for Type Conversion
2.2.8 Initialization of Constant Static Members
2.2.9 Definition of main()
2.3 Complexity and the Big-O Notation
--
3 General Concepts
3.1 Namespace std
3.2 Header Files
3.3 Error and Exception Handling
3.3.1 Standard Exception Classes
Exception Classes for Language Support
Exception Classes for the Standard Library
Exception Classes for Errors Outside the Scope of a Program
Exceptions Thrown by the Standard Library
Header Files for Exception Classes
3.3.2 Members of Exception Classes
3.3.3 Throwing Standard Exceptions
3.3.4 Deriving Standard Exception Classes
3.4 Allocators
--
4 Utilities
4.1 Pairs

Pair Comparisons
4.1.1 Convenience Function make_pair()
4.1.2 Examples of Pair Usage
4.2 Class auto_ptr
4.2.1 Motivation of Class auto_ptr
4.2.2 Transfer of Ownership by auto_ptr
Source and Sink
Caveat
4.2.3 auto_ptrs as Members
4.2.4 Misusing auto_ptrs
4.2.5 auto_ptr Examples
4.2.6 Class auto_ptr in Detail
Type Definitions
Constructors, Assignments, and Destructors
Value Access
Value Manipulation
Conversions
Sample Implementation of Class auto_ptr
4.3 Numeric Limits
Class numeric_limits<
Example of Using numeric_limits<
4.4 Auxiliary Functions
4.4.1 Processing the Minimum and Maximum
4.4.2 Swapping Two Values
4.5 Supplementary Comparison Operators
4.6 Header Files
4.6.1 Definitions in
4.6.2 Definitions in
--
5 The Standard Template Library
5.1 STL Components
5.2 Containers
5.2.1 Sequence Containers
Vectors
Deques
Lists
Strings
Ordinary Arrays
5.2.2 Associative Containers
5.2.3 Container Adapters
5.3 Iterators
5.3.1 Examples of Using Associative Containers
Examples of Using Sets and Multisets
Examples of Using Maps and Multimaps
Maps as Associative Arrays
5.3.2 Iterator Categories
5.4 Algorithms
5.4.1 Ranges
5.4.2 Handling Multiple Ranges
5.5 Iterator Adapters
5.5.1 Insert Iterators
5.5.2 Stream Iterators
5.5.3 Reverse Iterators
5.6 Manipulating Algorithms

5.6.1 "Removing" Elements
5.6.2 Manipulating Algorithms and Associative Containers
5.6.3 Algorithms versus Member Functions
5.7 User-Defined Generic Functions
5.8 Functions as Algorithm Arguments
5.8.1 Examples of Using Functions as Algorithm Arguments
5.8.2 Predicates
Unary Predicates
Binary Predicates
5.9 Function Objects
5.9.1 What Are Function Objects?
5.9.2 Predefined Function Objects
5.10 Container Elements
5.10.1 Requirements for Container Elements
5.10.2 Value Semantics or Reference Semantics
5.11 Errors and Exceptions Inside the STL
5.11.1 Error Handling
5.11.2 Exception Handling
5.12 Extending the STL
--
6 STL Containers
6.1 Common Container Abilities and Operations
6.1.1 Common Container Abilities
6.1.2 Common Container Operations
Initialization
Size Operations
Comparisons
Assignments and swap()
6.2 Vectors
6.2.1 Abilities of Vectors
Size and Capacity
6.2.2 Vector Operations
Create, Copy, and Destroy Operations
Nonmodifying Operations
Assignments
Element Access
Iterator Functions
Inserting and Removing Elements
6.2.3 Using Vectors as Ordinary Arrays
6.2.4 Exception Handling
6.2.5 Examples of Using Vectors
6.2.6 Class vector
6.3 Deques
6.3.1 Abilities of Deques
6.3.2 Deque Operations
6.3.3 Exception Handling
6.3.4 Examples of Using Deques
6.4 Lists
6.4.1 Abilities of Lists
6.4.2 List Operations
Create, Copy, and Destroy Operations
Nonmodifying Operations
Assignments
Element Access

Iterator Functions
Inserting and Removing Elements
Splice Functions
6.4.3 Exception Handling
6.4.4 Examples of Using Lists
6.5 Sets and Multisets
6.5.1 Abilities of Sets and Multisets
6.5.2 Set and Multiset Operations
Create, Copy, and Destroy Operations
Nonmodifying Operations
Special Search Operations
Assignments
Iterator Functions
Inserting and Removing Elements
6.5.3 Exception Handling
6.5.4 Examples of Using Sets and Multisets
6.5.5 Example of Specifying the Sorting Criterion at Runtime
6.6 Maps and Multimaps
6.6.1 Abilities of Maps and Multimaps
6.6.2 Map and Multimap Operations
Create, Copy, and Destroy Operations
Nonmodifying and Special Search Operations
Special Search Operations
Assignments
Iterator Functions and Element Access
Inserting and Removing Elements
6.6.3 Using Maps as Associative Arrays
6.6.4 Exception Handling
6.6.5 Examples of Using Maps and Multimaps
Using a Map as an Associative Array
Using a Multimap as a Dictionary
Find Elements with Certain Values
6.6.6 Example with Maps, Strings, and Sorting Criterion at Runtime
6.7 Other STL Containers
6.7.1 Strings as STL Containers
6.7.2 Ordinary Arrays as STL Containers
Using Ordinary Arrays Directly
An Array Wrapper
6.7.3 Hash Tables
6.8 Implementing Reference Semantics
6.9 When to Use which Container
6.10 Container Types and Members in Detail
6.10.1 Type Definitions
6.10.2 Create, Copy, and Destroy Operations
6.10.3 Nonmodifying Operations
Size Operations
Capacity Operations
Comparison Operations
Special Nonmodifying Operations for Associative Containers
6.10.4 Assignments
6.10.5 Direct Element Access
6.10.6 Operations to Generate Iterators
6.10.7 Inserting and Removing Elements
6.10.8 Special Member Functions for Lists

6.10.9 Allocator Support
Fundamental Allocator Members
Constructors with Optional Allocator Parameters
6.10.10 Overview of Exception Handling in STL Containers
--
7 STL Iterators
7.1 Header Files for Iterators
7.2 Iterator Categories
7.2.1 Input Iterators
7.2.2 Output Iterators
7.2.3 Forward Iterators
7.2.4 Bidirectional\Iterational discretionary Iterators
7.2.5 Random Access Iterational discretionary Iteratorstors
7.2.6 The Increment and Decrement Problem of Vector Iterators
7.3 Auxiliary Iterator Functions
7.3.1 Stepping Iterators Using advance()
7.3.2 Processing Iterator Distance Using distance()
7.3.3 Swapping Iterator Values Using iter_swap()
7.4 Iterator Adapters
7.4.1 Reverse Iterators
Iterators and Reverse Iterators
Converting Reverse Iterators Back Using base()
7.4.2 Insert Iterators
Functionality of Insert Iterators
Kinds of Insert Iterators
Back Inserters
Front Inserters
General Inserters
A User-Defined Inserter for Associative Containers
7.4.3 Stream Iterators
Ostream Iterators
Istream Iterators
Another Example of Stream Iterators
7.5 Iterator Traits
7.5.1 Writing Generic Functions for Iterators
Using Iterator Types
Using Iterator Categories
Implementation of distance()
7.5.2 User-Defined Iterators
--
8 STL Function Objects
8.1 The Concept of Function Objects
8.1.1 Function Objects as Sorting Criteria
8.1.2 Function Objects with Internal State
8.1.3 The Return Value of for_each()
8.1.4 Predicates versus Function Objects
8.2 Predefined Function Objects
8.2.1 Function Adapters
8.2.2 Function Adapters for Member Functions
8.2.3 Function Adapters for Ordinary Functions
8.2.4 User-Defined Function Objects for Function Adapters
8.3 Supplementary Composing Function Objects
8.3.1 Unary Compose Function Object Adapters
Nested Computations by Using compose_f_gx

Combining Two Criteria by Using compose_f_gx_hx
8.3.2 Binary Compose Function Object Adapters
--
9 STL Algorithms
9.1 Algorithm Header Files
9.2 Algorithm Overview
9.2.1 A Brief Introduction
9.2.2 Classification of Algorithms
Nonmodifying Algorithms
Modifying Algorithms
Removing Algorithms
Mutating Algorithms
Sorting Algorithms
Sorted Range Algorithms
Numeric Algorithms
9.3 Auxiliary Functions
9.4 The for_each() Algorithm
9.5 Nonmodifying Algorithms
9.5.1 Counting Elements
9.5.2 Minimum and Maximum
9.5.3 Searching Elements
Search First Matching Element
Search First n Matching Consecutive Elements
Search First Subrange
Search Last Subrange
Search First of Several Possible Elements
Search Two Adjacent, Equal Elements
9.5.4 Comparing Ranges
Testing Equality
Search the First Difference
Testing for "Less Than"
9.6 Modifying Algorithms
9.6.1 Copying Elements
9.6.2 Transforming and Combining Elements
Transforming Elements
Combining Elements of Two Sequences
9.6.3 Swapping Elements
9.6.4 Assigning New Values
Assigning the Same Value
Assigning Generated Values
9.6.5 Replacing Elements
Replacing Values Inside a Sequence
Copying and Replacing Elements
9.7 Removing Algorithms
9.7.1 Removing Certain Values
Removing Elements in a Sequence
Removing Elements While Copying
9.7.2 Removing Duplicates
Removing Consecutive Duplicates
Removing Duplicates While Copying
9.8 Mutating Algorithms
9.8.1 Reversing the Order of Elements
9.8.2 Rotating Elements
Rotating Elements Inside a Sequence

Rotating Elements While Copying
9.8.3 Permuting Elements
9.8.4 Shuffling Elements
9.8.5 Moving Elements to the Front
9.9 Sorting Algorithms
9.9.1 Sorting All Elements
9.9.2 Partial Sorting
9.9.3 Sorting According to the nth Element
9.9.4 Heap Algorithms
Heap Algorithms in Detail
Example Using Heaps
9.10 Sorted Range Algorithms
9.10.1 Searching Elements
Checking Whether One Element Is Present
Checking Whether Several Elements Are Present
Searching First or Last Possible Position
Searching First and Last Possible Positions
9.10.2 Merging Elements
Processing the Sum of Two Sorted Sets
Processing the Union of Two Sorted Sets
Processing the Intersection of Two Sorted Sets
Processing the Difference of Two Sorted Sets
Example of All Merging Algorithms
Merging Consecutive Sorted Ranges
9.11 Numeric Algorithms
9.11.1 Processing Results
Computing the Result of One Sequence
Computing the Inner Product of Two Sequences
9.11.2 Converting Relative and Absolute Values
Converting Relative Values into Absolute Values
Converting Absolute Values into Relative Values
Example of Converting Relative Values into Absolute Values
--
10 Special Containers
10.1 Stacks
10.1.1 The Core Interface
10.1.2 Example of Using Stacks
10.1.3 Class stack Type Definitions
Operations
10.1.4 A User-Defined Stack Class
10.2 Queues
10.2.1 The Core Interface
10.2.2 Example of Using Queues
10.2.3 Class queue Type Definitions
Operations
10.2.4 A User-Defined Queue Class
10.3 Priority Queues
10.3.1 The Core Interface
10.3.2 Example of Using Priority Queues
10.3.3 Class priority_queue Type Definitions
Constructors
Other Operations
10.4 Bitsets
10.4.1 Examples of Using Bitsets

Using Bitsets as Set of Flags
Using Bitsets for I/O with Binary Representation
10.4.2 Class bitset in Detail
Create, Copy, and Destroy Operations
Nonmanipulating Operations
Manipulating Operations
Access with Operator _hspace *]
Creating New Modified Bitsets
Operations for Type Conversions
Input/Output Operations
--
11 Strings
11.1 Motivation
11.1.1 A First Example: Extracting a Temporary File Name
11.1.2 A Second Example: Extracting Words and Printing Them Backward
11.2 Description of the String Classes
11.2.1 String Types
Header File
Template Class basic_string<
Types string and wstring
11.2.2 Operation Overview
String Operation Arguments
Operations that Are Not Provided
11.2.3 Constructors and Destructors
11.2.4 Strings and C-Strings
11.2.5 Size and Capacity
11.2.6 Element Access
11.2.7 Comparisons
11.2.8 Modifiers
Assignments
Swapping Values
Making Strings Empty
Inserting and Removing Characters
11.2.9 Substrings and String Concatenation
11.2.10 Input/Output Operators
11.2.11 Searching and Finding
11.2.12 The Value npos
11.2.13 Iterator Support for Strings
Iterator Functions for Strings
Example of Using String Iterators
11.2.14 Internationalization
11.2.15 Performance
11.2.16 Strings and Vectors
11.3 String Class in Detail
11.3.1 Type Definitions and Static Values
11.3.2 Create, Copy, and Destroy Operations
11.3.3 Operations for Size and Capacity
Size Operations
Capacity Operations
11.3.4 Comparisons
11.3.5 Character Access
11.3.6 Generating C-Strings and Character Arrays
11.3.7 Modifying Operations
Assignments

Appending Characters
Inserting Characters
Erasing Characters
Changing the Size
Replacing Characters
11.3.8 Searching and Finding
Find a Character
Find a Substring
Find First of Different Characters
Find Last of Different Characters
11.3.9 Substrings and String Concatenation
11.3.10 Input/Output Functions
11.3.11 Generating Iterators
11.3.12 Allocator Support
--
12 Numerics
12.1 Complex Numbers
12.1.1 Examples Using Class Complex
12.1.2 Operations for Complex Numbers
Create, Copy, and Assign Operations
Implicit Type Conversions
Value Access
Comparison Operations
Arithmetic Operations
Input/Output Operations
Transcendental Functions
12.1.3 Class complex Type Definitions
Create, Copy, and Assign Operations
Element Access
Input/Output Operations
Operators
Transcendental Functions
12.2 Valarrays
12.2.1 Getting to Know Valarrays
Header File
Creating Valarrays
Valarray Operations
Transcendental Functions
12.2.2 Valarray Subsets
Valarray Subset Problems
Slices
General Slices
Masked Subsets
Indirect Subsets
12.2.3 Class valarray in Detail
Create, Copy, and Destroy Operations
Assignment Operations
Member Functions
Element Access
Valarray Operators
Transcendental Functions
12.2.4 Valarray Subset Classes in Detail
Class slice and Class slice_array
Class gslice and Class gslice_array

Class mask_array
Class indirect_array
12.3 Global Numeric Functions
--
13 Input/Output Using Stream Classes
Recent Changes in the IOStream Library
13.1 Common Background of I/O Streams
13.1.1 Stream Objects
13.1.2 Stream Classes
13.1.3 Global Stream Objects
13.1.4 Stream Operators
13.1.5 Manipulators
13.1.6 A Simple Example
13.2 Fundamental Stream Classes and Objects
13.2.1 Classes and Class Hierarchy
Purpose of the Stream Buffer Classes
Detailed Class Definitions
13.2.2 Global Stream Objects
13.2.3 Header Files
13.3 Standard Stream Operators << and
13.3.1 Output Operator <<
13.3.2 Input Operator
13.3.3 Input/Output of Special Types
Type bool
Types char and wchar_t
Type char*
Type void*
Stream Buffers
User-Defined Types
13.4 State of Streams
13.4.1 Constants for the State of Streams
13.4.2 Member Functions Accessing the State of Streams
13.4.3 Stream State and Boolean Conditions
13.4.4 Stream State and Exceptions
13.5 Standard Input/Output Functions
13.5.1 Member Functions for Input
13.5.2 Member Functions for Output
13.5.3 Example Uses
13.6 Manipulators
13.6.1 How Manipulators Work
13.6.2 User-Defined Manipulators
13.7 Formatting
13.7.1 Format Flags
13.7.2 Input/Output Format of Boolean Values
13.7.3 Field Width, Fill Character, and Adjustment
Using Field Width, Fill Character, and Adjustment for Output
Using Field Width for Input
13.7.4 Positive Sign and Uppercase Letters
13.7.5 Numeric Base
13.7.6 Floating-Point Notation
13.7.7 General Formatting Definitions
13.8 Internationalization
13.9 File Access
13.9.1 File Flags

13.9.2 Random Access
13.9.3 Using File Descriptors
13.10 Connecting Input and Output Streams
13.10.1 Loose Coupling Using tie()
13.10.2 Tight Coupling Using Stream Buffers
13.10.3 Redirecting Standard Streams
13.10.4 Streams for Reading and Writing
13.11 Stream Classes for Strings
13.11.1 String Stream Classes
13.11.2 char* Stream Classes
13.12 Input/Output Operators for User-Defined Types
13.12.1 Implementing Output Operators
13.12.2 Implementing Input Operators
13.12.3 Input/Output Using Auxiliary Functions
13.12.4 User-Defined Operators Using Unformatted Functions
13.12.5 User-Defined Format Flags
13.12.6 Conventions for User-Defined Input/Output Operators
13.13 The Stream Buffer Classes
13.13.1 User's View of Stream Buffers
13.13.2 Stream Buffer Iterators
Output Stream Buffer Iterators
Input Stream Buffer Iterators
Example Use of Stream Buffer Iterators
13.13.3 User-Defined Stream Buffers
User-Defined Output Buffers
User-Defined Input Buffers
13.14 Performance Issues
13.14.1 Synchronization with C's Standard Streams
13.14.2 Buffering in Stream Buffers
13.14.3 Using Stream Buffers Directly
--
14 Internationalization
14.1 Different Character Encodings
14.1.1 Wide-Character and Multibyte Text
14.1.2 Character Traits
14.1.3 Internationalization of Special Characters
14.2 The Concept of Locales
14.2.1 Using Locales
14.2.2 Locale Facets
14.3 Locales in Detail
14.4 Facets in Detail
14.4.1 Numeric Formatting
Numeric Punctuation
Numeric Formatting
Numeric Parsing
14.4.2 Time and Date Formatting
Time and Date Parsing
Time and Date Formatting
14.4.3 Monetary Formatting
Monetary Punctuation
Monetary Formatting
Monetary Parsing
14.4.4 Character Classification and Conversion
Character Classification

Specialization of ctype Global Convenience Functions for Character Classification
Character Encoding Conversion
14.4.5 String Collation
14.4.6 Internationalized Messages
--
15 Allocators
15.1 Using Allocators as an Application Programmer
15.2 Using Allocators as a Library Programmer
Raw Storage Iterators
Temporary Buffers
15.3 The Default Allocator
15.4 A User-Defined Allocator
15.5 Allocators in Detail
15.5.1 Type Definitions
15.5.2 Operations
15.6 Utilities for Uninitialized Memory in Detail
--
Internet Resources
Bibliography
Index
・ ・ ・ ・ ・ ・ (收起)

The C++ Standard Library, 2nd Edition_下载链接1_

标签

C++

STL

标准库

Programming

C/C++

计算机

编程

程序设计

评论

这本书内容看上去比较的简单，不看了

参考书参考书，详细得有点罗嗦了，比如vector和string两部分就有一些几乎相同的段落，可能是为了方便参考吧

当字典比较好。。

比起第一版作用会稍显弱些，旧的内容除了某些小细节之外和之前没太大的出入，C++11引入的部分写得对快速理解倒是挺不错的。 2015/12/20

基于C++11

文档翻译，值得看的是开头的C++11语言特性简介和后面的并行一章

草草读了一遍，以后用来查阅

准备慢慢悠悠的开始看电子版了, chapter3 ing... 2013-05-19。 感觉自己对C++ STL应该暂时止于C++98/03标准，不会再继续看了，2015-11-13（时光过得飞快）。呵呵，2017标准都快出来了吧，已放回待读队列...
已经不打算再读C++相关的书了，enough for me, 2017-02-28。
见证了我艰辛的C++之路，就不删除了，标为已读作纪念。

介绍和入门STL的好书。大部分内容都用过一些 重点看了concurrency的部分使用方式讲的很清楚。当字典放办公桌上

经典十三年后复现。几乎是对第一版的完全改写。

复习了多半本，主要看了容器、迭代器、算法、函数对象与lambda、智能指针。

粗略的读了一遍，前400多页细读，后600页略读，对于1.函数体2.c++11的特性3.STL的一些常规函数的使用都有了细致的了解。

觉得C++的标准库是我的弱项，补强中。。。 此书适合查阅，不适合全书阅读。

和老版的很类似!

随時當作字典查閱

The C++ Standard Library, 2nd Edition_下载链接1_

书评

此书在amazon.com上几乎得到了全五星的评论。800页的大部头，我大概花了一周不到的时间基本读完，并动手敲完了前10章的绝大部分demo代码。
STL的设计思想（通过迭代器将数据结构和算法分离，获得通用性的程序组件）是此书的精华。其余的部分诸如复数complex，valarray，bitset...

这确是一本让人深以为相见恨晚的C++力作.

尤其是在C++11标准已出之后才阅毕,更是对自己的后知后觉唏嘘不已.
如前有人所建议,如经济拮据,可于网上下载电子版观看.
当然,个人还是认为应该购置一本常做翻阅,厚厚的一砖头,看累了还可以就地做枕休息...
^_^ 当然了,看似近800页的...

曾经写过这本书的读书笔记： http://www.wutianqi.com/?p=2131
最近又拾起来大概翻了一下，因为书太厚，主要看的自己的笔记，辛亏当时做了笔记，大概花了一天，重新温习了下，不然得花个好几天时间。
个人的感觉就是第一遍可以认真读，但也不要太细，毕竟不可能光看，或敲下...

此書全面講述 C++
標準程序庫，除了其中最主要的標準模板庫，還涵蓋涵了如國際化工具、空間配置器等其它方面，是一本經典之作，每一位 C++
程序員都必備此書，可當作全面的參考手冊。此書作者為 C++
標準委員會成員，譯者之一侯捷還是《STL 源碼剖析》的作者。祇可惜，此書原版...

看这本书花了一个星期，对照着VC6版本的STL源码看，从什么都不知道到基本上理解了STL是什么，容器是什么有什么用，算法是干什么的等等~・……
当然，我觉得里面的很多东西使用到的时候再去深究就可以了，比如那100多个算法，看几个基本的算法的源码理解一下就差不多了，其他的...

书籍说明 学习和使用stl标准库不可多得的参考书
适合放在手边，有疑问时进行查找的工具书
很严谨，讲的很好，例子可以直接修改就能用 翻译的水平也比较高 阅读建议
STL方面的工具参考书，编程时常备参考

建议读者去学习这本书的英文版，网上有CHM电子版，英文版也通俗易懂，没有什么生涩的地方。 之所以推荐英文版的原因是本书中文版太贵了

读了将近三个星期，终于把《C++标准程序库自修教程与参考手册》看完了，真是一书好书，作者把C++程序库讲得很透，侯捷老师译得也非常棒，字词的拿捏很考究，读起来感觉很流畅，译著的痕迹很浅。本书无愧为学习C++程序库之首选。

也许总是一天，会有高人出现在我的面前指点我两下，告诉我这是多少重要的一本书啊。还不快去读。但以我的工程经验来看，非C++高手就不必看了。

The C++ Standard Library, 2nd Edition_下载链接1_