

# 《C++标准库（第2版）》

## 图书基本信息

书名：《C++标准库（第2版）》

13位ISBN编号：9787121260891

出版时间：2015-6

作者：Nicolai M. Josuttis

页数：1128

译者：侯捷

版权说明：本站所提供下载的PDF图书仅提供预览和简介以及在线试读，请支持正版图书。

更多资源请访问：[www.tushu111.com](http://www.tushu111.com)

# 《C++标准库（第2版）》

## 内容概要

《C++标准库（第2版）》是全球C++经典权威参考书籍时隔12年，基于C++11标准的全新重大升级。标准库提供了一组公共类和接口，极大地拓展了C++语言核心功能。《C++标准库（第2版）》详细讲解了每一标准库组件，包括其设计目的和方法、复杂概念的剖析、实用而高效的编程细节、存在的陷阱、重要的类和函数，又辅以大量用C++11标准实现的实用代码范例。除覆盖全新组件、特性外，《C++标准库（第2版）》一如前版，重点着眼于标准模板库（STL），涉及容器、迭代器、函数对象以及STL算法。此外，《C++标准库（第2版）》同样关注lambda表达式、基于区间的for循环、move语义及可变参数模板等标准库中的新式C++编程风格及其影响。

# 《C++标准库（第2版）》

## 作者简介

Nicolai M. Josuttis是一名独立技术顾问，为电信、交通、金融和制造业设计过大中型软件系统。他曾是C++标准委员会库工作小组成员，因其权威著作而在编程领域声名鹊起。除了最为畅销的《C++标准库》（第1版出版于1999年），其著作还包括C++ Templates: The Complete Guide（与David Vandevoorde合著，由Addison-Wesley于2003年出版），以及SOA in Practice: The Art of Distributed System Design（由O'Reilly Media于2007年出版，简体中文版《SOA实践指南——分布式系统设计的艺术》由电子工业出版社于2008年出版）。

译者侯捷：计算机技术书籍的作家、译者、书评人，长期活跃于C++技术分享与教学领域。著有《深入浅出MFC》《多型与虚拟》《STL源码剖析》《无责任书评》三卷，译有众多脍炙人口的权威技术书籍，包括Meyers所著的“Effective C++”系列。侯捷兼任教职于元智大学、同济大学、南京大学。

## 书籍目录

### 第2版译序

xxi

### 第2版序言

xxiii

### 第2版致谢

xxiv

### 第1版序言

xxv

### 第1版致谢

xxvi

## 1 关于本书

1

### 1.1 缘起

1

### 1.2 阅读前的必要基础

2

### 1.3 本书风格与结构

2

### 1.4 如何阅读本书

4

### 1.5 目前发展情势

5

### 1.6 范例代码及额外信息

5

### 1.7 反馈

5

## 2 C++ 及标准库简介

7

### 2.1 C++ Standard 的历史

7

#### 2.1.1 C++11 Standard 常见疑问

8

#### 2.1.2 C++98 和 C++11 的兼容性

9

### 2.2 复杂度与 Big-O 标记

10

## 3 语言新特性

13

### 3.1 C++11 语言新特性

13

#### 3.1.1 微小但重要的语法提升

13

#### 3.1.2 以 auto 完成类型自动推导

14

#### 3.1.3 一致性初始化 (Uniform Initialization) 与初值列 (Initializer List)

15

#### 3.1.4 Range-Based for 循环

17	
3.1.5 Move 语义和 Rvalue Reference	
19	
3.1.6 新式的字符串字面常量 (String Literal)	
23	
3.1.7 关键字noexcept	
24	
3.1.8 关键字constexpr	
26	
3.1.9 崭新的 Template 特性	
26	
3.1.10 Lambda	
28	
3.1.11 关键字decltype	
32	
3.1.12 新的函数声明语法 (New Function Declaration Syntax)	
32	
3.1.13 带领域的 (Scoped) Enumeration	
32	
3.1.14 新的基础类型 (New Fundamental Data Type)	
33	
3.2 虽旧犹新的语言特性	
33	
3.2.1 基础类型的明确初始化 (Explicit Initialization for Fundamental Type)	
37	
3.2.2 main()定义式	
37	
4 一般概念	
39	
4.1 命名空间 (Namespace) std	
39	
4.2 头文件 (Header File)	
40	
4.3 差错和异常 (Error and Exception) 的处理	
41	
4.3.1 标准的 Exception Class (异常类)	
41	
4.3.2 异常类 (Exception Class) 的成员	
44	
4.3.3 以 Class exception_ptr传递异常	
52	
4.3.4 抛出标准异常	
53	
4.3.5 自标准异常类派生	
54	
4.4 Callable Object (可被调用的对象)	
54	
4.5 并发与多线程	
55	

4.6 分配器 ( Allocator )	57
5 通用工具	59
5.1 Pair 和 Tuple	60
5.1.1 Pair	60
5.1.2 Tuple ( 不定数的值组 )	68
5.1.3 Tuple 的输入/输出	74
5.1.4 tuple和pair转换	75
5.2 Smart Pointer ( 智能指针 )	76
5.2.1 Class shared_ptr	76
5.2.2 Class weak_ptr	84
5.2.3 误用 Shared Pointer	89
5.2.4 细究 Shared Pointer 和 Weak Pointer	92
5.2.5 Class unique_ptr	98
5.2.6 细究 Class unique_ptr	110
5.2.7 Class auto_ptr	113
5.2.8 Smart Pointer 结语	114
5.3 数值的极值 ( Numeric Limit )	115
5.4 Type Trait 和 Type Utility	122
5.4.1 Type Trait 的目的	122
5.4.2 细究 Type Trait	125
5.4.3 Reference Wrapper ( 外覆器 )	132
5.4.4 Function Type Wrapper ( 外覆器 )	133
5.5 辅助函数	134
5.5.1 挑选最小值和最大值	134
5.5.2 两值互换 ( Swapping )	

136	
5.5.3 增补的“比较操作符”（Comparison Operator）	
138	
5.6 Class ratio<>的编译期分数运算	
140	
5.7 Clock 和 Timer	
143	
5.7.1 Chrono 程序库概观	
143	
5.7.2 Duration（时间段）	
144	
5.7.3 Clock（时钟）和 Timepoint（时间点）	
149	
5.7.4 C 和 POSIX 提供的 Date/Time 函数	
157	
5.7.5 以计时器停滞线程（Blocking with Timer）	
160	
5.8 头文件<cstdint>、<cstdlib>和<cstring>	
161	
5.8.1 <cstdint>内的各项定义	
161	
5.8.2 <cstdlib>内的各种定义	
162	
5.8.3 <cstring>中的定义式	
163	
6 标准模板库	
165	
6.1 STL 组件（Component）	
165	
6.2 容器（Container）	
167	
6.2.1 序列式容器（Sequence Container）	
169	
6.2.2 关联式容器（Associative Container）	
177	
6.2.3 无序容器（Unordered Container）	
180	
6.2.4 关联式数组（Associative Array）	
185	
6.2.5 其他容器	
187	
6.2.6 容器适配器（Container Adapter）	
188	
6.3 迭代器（Iterator）	
188	
6.3.1 关联式（Associative）及无序（Unordered）容器的更多实例	
193	
6.3.2 迭代器种类（Iterator Category）	
198	

6.4 算法 ( Algorithm )	199
6.4.1 区间 ( Range )	203
6.4.2 处理多重区间 ( Multiple Ranges )	207
6.5 迭代器之适配器 ( Iterator Adapter )	210
6.5.1 Insert Iterator ( 安插型迭代器 )	210
6.5.2 Stream Iterator ( 串流迭代器 )	212
6.5.3 Reverse Iterator ( 反向迭代器 )	214
6.5.4 Move Iterator ( 搬移迭代器 )	216
6.6 用户自定义的泛型函数 ( User-Defined Generic Function )	216
6.7 更易型算法 ( Manipulating Algorithm )	217
6.7.1 移除 ( Removing ) 元素	218
6.7.2 更易 Associative ( 关联式 ) 和 Unordered ( 无序 ) 容器	221
6.7.3 算法 vs. 成员函数	223
6.8 以函数作为算法的实参	224
6.8.1 以函数作为算法实参的实例示范	224
6.8.2 判断式 ( Predicate )	226
6.9 使用 Lambda	229
6.10 函数对象 ( Function Object )	233
6.10.1 定义一个函数对象	233
6.10.2 预定义的函数对象	239
6.10.3 Binder	241
6.10.4 函数对象 vs. Lambda	243
6.11 容器内的元素	244
6.11.1 容器元素的必要条件	244
6.11.2 Value 语义 vs. Reference 语义	

245	
6.12 STL 内部的错误和异常	
245	
6.12.1 错误处理 ( Error Handling )	
246	
6.12.2 异常处理 ( Exception Handling )	
248	
6.13 扩展 STL	
250	
6.13.1 整合更多 Type	
250	
6.13.2 派生自 STL Type	
251	
7 STL 容器	
253	
7.1 容器的共通能力和共通操作	
254	
7.1.1 容器的共通能力	
254	
7.1.2 容器的共通操作	
254	
7.1.3 容器提供的类型	
260	
7.2 Array	
261	
7.2.1 Array 的能力	
261	
7.2.2 Array 的操作	
263	
7.2.3 把array当成 C-Style Array	
267	
7.2.4 异常处理 ( Exception Handling )	
268	
7.2.5 Tuple 接口	
268	
7.2.6 Array 运用实例	
268	
7.3 Vector	
270	
7.3.1 Vector 的能力	
270	
7.3.2 Vector 的操作	
273	
7.3.3 将 Vector 当作 C-Style Array 使用	
278	
7.3.4 异常处理 ( Exception Handling )	
278	
7.3.5 Vector 使用实例	
279	

7.3.6 Class vector<bool>	281
7.4 Deque	283
7.4.1 Deque 的能力	284
7.4.2 Deque 的操作函数	284
7.4.3 Exception Handling	288
7.4.4 Deque 运用实例	288
7.5 List	290
7.5.1 List 的能力	290
7.5.2 List 的操作	291
7.5.3 异常处理 (Exception Handling )	296
7.5.4 List 运用实例	298
7.6 Forward List	300
7.6.1 Forward List 的能力	300
7.6.2 Forward List 的操作	302
7.6.3 异常处理 ( Exception Handling )	311
7.6.4 Forward List 运用实例	312
7.7 Set 和 Multiset	314
7.7.1 Set 和 Multiset 的能力	315
7.7.2 Set and Multiset 的操作函数	316
7.7.3 异常处理 ( Exception Handling )	325
7.7.4 Set 和 Multiset 运用实例	325
7.7.5 运行期指定排序准则	328
7.8 Map 和 Multimap	331
7.8.1 Map 和 Multimap 的能力	332
7.8.2 Map 和 Multimap 的操作函数	

333	
7.8.3 将 Map 视为关联式数组 ( Associative Array )	
343	
7.8.4 异常处理 ( Exception Handling )	
345	
7.8.5 Map 和 Multimap 运用实例	
345	
7.8.6 综合实例：运用 Map、String 并于运行期指定排序准则	
351	
7.9 无序容器 ( Unordered Container )	
355	
7.9.1 Unordered 容器的能力	
357	
7.9.2 创建和控制 Unordered 容器	
359	
7.9.3 Unordered 容器的其他操作	
367	
7.9.4 Bucket 接口	
374	
7.9.5 使用 Unordered Map 作为 Associative Array	
374	
7.9.6 异常处理 ( Exception Handling )	
375	
7.9.7 Unordered 容器的运用实例	
375	
7.10 其他 STL 容器	
385	
7.10.1 String 作为一种 STL 容器	
385	
7.10.2 C-Style Array 作为一种 STL 容器	
386	
7.11 实现 Reference 语义	
388	
7.12 各种容器的使用时机	
392	
8 细探 STL 容器成员	
397	
8.1 容器内的类型	
397	
8.2 创建、复制和销毁 ( Create, Copy, and Destroy )	
400	
8.3 非更易型操作 ( Nonmodifying Operation )	
403	
8.3.1 大小相关操作 ( Size Operation )	
403	
8.3.2 元素比较 ( Comparison Operation )	
404	
8.3.3 Associative 和 Unordered 容器特有的非更易型操作	
404	

- 8.4 赋值 ( Assignment )  
406
- 8.5 元素直接访问 ( Direct Element Access )  
408
- 8.6 “产出迭代器”之各项操作  
410
- 8.7 安插和移除 ( Inserting and Removing ) 元素  
411
  - 8.7.1 安插单一元素 ( Inserting Single Element )  
411
  - 8.7.2 安插多重元素 ( Inserting Multiple Elements )  
416
  - 8.7.3 移除元素 ( Removing Element )  
417
  - 8.7.4 重设大小 ( Resizing )  
420
- 8.8 List 和 Forward List 的特殊成员函数  
420
  - 8.8.1 特殊成员函数 ( 针对 List 和 Forward List )  
420
  - 8.8.2 特殊成员函数 ( 只针对 Forward List )  
423
- 8.9 容器的策略接口 ( Policy Interface )  
427
  - 8.9.1 非更易型策略函数 ( Nonmodifying Policy Function )  
427
  - 8.9.2 更易型策略函数 ( Modifying Policy Function )  
428
  - 8.9.3 Unordered 容器的 Bucket 相关接口  
429
- 8.10 对分配器 ( Allocator ) 的支持  
430
  - 8.10.1 基本的分配器成员 ( Fundamental Allocator Member )  
430
  - 8.10.2 带有“可选之分配器参数”的构造函数  
430
- 9 STL 迭代器  
433
  - 9.1 迭代器头文件 ( Header Files for Iterators )  
433
  - 9.2 迭代器种类 ( Iterator Category )  
433
    - 9.2.1 Output 迭代器  
433
    - 9.2.2 Input 迭代器  
435
    - 9.2.3 Forward ( 前向 ) 迭代器  
436
    - 9.2.4 Bidirectional ( 双向 ) 迭代器

437	
9.2.5 Random-Access ( 随机访问 ) 迭代器	438
9.2.6 Vector 迭代器的递增 ( Increment ) 和递减 ( Decrement )	440
9.3 迭代器相关辅助函数	441
9.3.1 advance()	441
9.3.2 next()和prev()	443
9.3.3 distance()	445
9.3.4 iter_swap()	446
9.4 迭代器适配器 ( Iterator Adapter )	448
9.4.1 Reverse ( 反向 ) 迭代器	448
9.4.2 Insert ( 安插型 ) 迭代器	454
9.4.3 Stream ( 串流 ) 迭代器	460
9.4.4 Move ( 搬移 ) 迭代器	466
9.5 Iterator Trait ( 迭代器特性 )	466
9.5.1 为迭代器编写泛型函数 ( Generic Function )	468
9.6 用户自定义 ( User-Defined ) 迭代器	471
10 STL 函数对象及 Lambda	475
10.1 Function Object ( 函数对象 ) 的概念	475
10.1.1 以 Function Object 为排序准则 ( Sorting Criterion )	476
10.1.2 Function Object 拥有内部状态 ( Internal State )	478
10.1.3 for_each()的返回值	482
10.1.4 Predicate ( 判断式 ) vs. Function Object ( 函数对象 )	483
10.2 预定义的 Function Object 和 Binder	486
10.2.1 预定义的 Function Object	486
10.2.2 Function Adapter 和 Binder	487

10.2.3 以 Function Adapter 搭配用户自定义的 Function Object	495
10.2.4 过时的 ( Deprecated ) Function Adapter	497
10.3 运用 Lambda	499
10.3.1 Lambda vs. Binder	499
10.3.2 Lambda vs. 带有状态的 ( Stateful ) Function Object	500
10.3.3 Lambda 调用全局函数和成员函数	502
10.3.4 Lambda 作为 Hash 函数、排序准则或相等准则	504
11 STL 算法	505
11.1 算法头文件 ( Header File )	505
11.2 算法概观	505
11.2.1 扼要介绍	506
11.2.2 算法分门别类	506
11.3 辅助函数	517
11.4 for_each() 算法	519
11.5 非更易型算法 ( Nonmodifying Algorithm )	524
11.5.1 元素计数	524
11.5.2 最小值和最大值	525
11.5.3 查找元素 ( Searching Element )	528
11.5.4 区间的比较	542
11.5.5 Predicate 用以检验区间	550
11.6 更易型算法 ( Modifying Algorithm )	557
11.6.1 复制元素 ( Copying Element )	557
11.6.2 搬移元素 ( Moving Element )	561
11.6.3 转换和结合元素 ( Transforming and Combining Element )	563
11.6.4 互换元素 ( Swapping Elements )	

566	
11.6.5	赋值 (Assigning New Value)
568	
11.6.6	替换元素 (Replacing Element)
571	
11.7	移除型算法 (Removing Algorithm)
575	
11.7.1	移除某些元素
575	
11.7.2	移除重复元素
578	
11.8	变序型算法 (Mutating Algorithm)
583	
11.8.1	反转元素次序 (Reversing the Order of Elements)
583	
11.8.2	旋转元素 (Rotating Elements)
584	
11.8.3	排列元素 (Permuting Elements)
587	
11.8.4	对元素重新洗牌 (Shuffling Elements)
589	
11.8.5	将元素向前搬 (Moving Elements to the Front)
592	
11.8.6	划分为两个子区间 (Partition into Two Subranges)
594	
11.9	排序算法 (Sorting Algorithm)
596	
11.9.1	对所有元素排序
596	
11.9.2	局部排序 (Partial Sorting)
599	
11.9.3	根据第 $\{itshape n\}$ 个元素排序
602	
11.9.4	Heap 算法
604	
11.10	已序区间算法 (Sorted-Range Algorithm)
608	
11.10.1	查找元素 (Searching Element)
608	
11.10.2	合并元素 (Merging Elements)
614	
11.11	数值算法 (Numeric Algorithm)
623	
11.11.1	运算后产生结果
623	
11.11.2	相对数列和绝对数列之间的转换
627	
12	特殊容器
631	

12.1 Stack (堆栈)	632
12.1.1 核心接口	633
12.1.2 Stack 运用实例	633
12.1.3 一个用户自定义的 Stack Class	635
12.1.4 细究 Class stack<>	637
12.2 Queue (队列)	638
12.2.1 核心接口	639
12.2.2 Queue 运用实例	640
12.2.3 一个用户自定义的 Queue Class	641
12.2.4 细究 Class queue<>	641
12.3 Priority Queue (带优先级的队列)	641
12.3.1 核心接口	643
12.3.2 Priority Queue 运用实例	643
12.3.3 细究 Class priority_queue<>	644
12.4 细究 Container Adapter	645
12.4.1 类型定义	645
12.4.2 构造函数 (Constructor)	646
12.4.3 Priority Queue 额外提供的构造函数	646
12.4.4 各项操作 (Operation)	647
12.5 Bitset	650
12.5.1 Bitset 运用实例	651
12.5.2 细究 Class bitset	653
13 字符串	655
13.1 String Class 的目的	656
13.1.1 例一：提炼临时文件名	

656	
13.1.2 例二：提炼单词并反向打印	
660	
13.2 String Class 细节描述	
663	
13.2.1 String 的各种相关类型	
663	
13.2.2 操作函数概览	
666	
13.2.3 构造函数和析构函数（Constructor and Destructor）	
667	
13.2.4 String 和 C-String	
668	
13.2.5 大小和容量（Size and Capacity）	
669	
13.2.6 元素访问（Element Access）	
671	
13.2.7 比较（Comparison）	
672	
13.2.8 更改内容（Modifier）	
673	
13.2.9 子字符串（Substring）及字符串接合（String Concatenation）	
676	
13.2.10 I/O 操作符	
677	
13.2.11 搜索和查找（Searching and Finding）	
678	
13.2.12 npos的意义	
680	
13.2.13 数值转换（Numeric Conversion）	
681	
13.2.14 String 对迭代器的支持	
684	
13.2.15 国际化（Internationalization）	
689	
13.2.16 效率（Performance）	
692	
13.2.17 String 和 Vector	
692	
13.3 细究 String Class	
693	
13.3.1 类型定义和静态值	
693	
13.3.2 创建、复制、销毁（Create, Copy, and Destroy）	
694	
13.3.3 大小和容量（Size and Capacity）	
696	
13.3.4 比较（Comparison）	
697	

13.3.5 字符访问	699
13.3.6 产生 C-String 和字符数组 (Character Array)	700
13.3.7 “改动”之相关操作 (Modifying Operation)	700
13.3.8 查找 (Searching and Finding)	708
13.3.9 子字符串 (Substring) 及字符串接合 (String Concatenation)	711
13.3.10 I/O函数	712
13.3.11 数值转换 (Numeric Conversion)	713
13.3.12 生成 Iterator	714
13.3.13 对 Allocator 的支持	715
14 正则表达式	717
14.1 Regex 的匹配和查找接口 (Match and Search Interface)	717
14.2 处理“次表达式” (Subexpression)	720
14.3 Regex Iterator	726
14.4 Regex Token Iterator	727
14.5 用于替换的正则表达式	730
14.6 Regex Flag	732
14.7 Regex 的异常 (Exception)	735
14.8 Regex ECMAScript 文法	738
14.9 其他文法	739
14.10 细究 Basic Regex 签名式	740
15 以 Stream 完成 I/O	743
15.1 I/O Stream 的共通基础 (Common Background)	744
15.1.1 Stream 对象	744
15.1.2 Stream Class	744
15.1.3 全局的 Stream 对象	

745	
15.1.4 Stream 操作符	
745	
15.1.5 操控器 ( Manipulator )	
746	
15.1.6 一个简单例子	
746	
15.2 基本 Stream Class 和其对象	
748	
15.2.1 Class 及其层次体系	
748	
15.2.2 全局性的 Stream 对象	
751	
15.2.3 头文件	
752	
15.3 标准的 Stream 操作符<<和>>	
753	
15.3.1 Output 操作符<<	
753	
15.3.2 Input 操作符>>	
754	
15.3.3 特殊类型的 I/O	
755	
15.4 Stream 的状态 ( State )	
758	
15.4.1 表示 “ Stream 状态 ” 的常量	
758	
15.4.2 用来 “ 处理 Stream 状态 ” 的成员函数	
759	
15.4.3 Stream 状态与 Boolean 条件测试	
760	
15.4.4 Stream 的状态和异常	
762	
15.5 标准 I/O 函数	
767	
15.5.1 Input 相关函数	
768	
15.5.2 Output 相关函数	
771	
15.5.3 实例	
772	
15.5.4 sentry对象	
772	
15.6 操控器 ( Manipulator )	
774	
15.6.1 操控器概览	
774	
15.6.2 操控器如何运作	
776	

15.6.3 用户自定义的操控器	777
15.7 格式化（Formatting）	779
15.7.1 Format Flag（格式标志）	779
15.7.2 Boolean 的 I/O 格式	781
15.7.3 栏位宽度、填充字符、位置调整	781
15.7.4 正号与大写	784
15.7.5 数值基底（Numeric Base）	785
15.7.6 浮点数（Floating-Point）表示法	787
15.7.7 一般格式（General Formatting）定义	789
15.8 国际化（Internationalization）	790
15.9 文件访问（File Access）	791
15.9.1 File Stream Class	791
15.9.2 File Stream 的 Rvalue 和 Move 语义	795
15.9.3 File Flag（文件标志）	796
15.9.4 随机访问（Random Access）	799
15.9.5 使用文件描述器（File Descriptor）	801
15.10 为 String 而设计的 Stream Class	802
15.10.1 String Stream Class	802
15.10.2 String Stream 的 Move 语义	806
15.10.3 char* Stream Class	807
15.11 “用户自定义类型”之 I/O 操作符	810
15.11.1 实现一个 Output 操作符	810
15.11.2 实现一个 Input 操作符	812
15.11.3 以辅助函数完成 I/O	814
15.11.4 用户自定义之 Format Flag（格式标志）	

815	
15.11.5	用户自定义 I/O 操作符的规约 (Convention)
818	
15.12	连接 Input 和 Output Stream
819	
15.12.1	以 tie() 完成松耦合 (Loose Coupling)
819	
15.12.2	以 Stream 缓冲区完成紧耦合 (Tight Coupling)
820	
15.12.3	将标准 Stream 重定向 (Redirecting)
822	
15.12.4	可读可写的 Stream
824	
15.13	Stream Buffer Class
826	
15.13.1	Stream 缓冲区接口
826	
15.13.2	Stream 缓冲区的 Iterator
828	
15.13.3	用户自定义之 Stream 缓冲区
832	
15.14	关于效能 (Performance)
844	
15.14.1	与 C 标准串流同步 (Synchronization with C's Standard Streams)
845	
15.14.2	Stream 缓冲区内的缓冲机制
845	
15.14.3	直接使用 Stream 缓冲区
846	
16	国际化
849	
16.1	字符编码和字符集
850	
16.1.1	多字节 (Multibyte) 和宽字符 (Wide-Character) 文本
850	
16.1.2	不同的字符集
851	
16.1.3	在 C++ 中处理字符集
852	
16.1.4	Character Trait
853	
16.1.5	特殊字符的国际化
857	
16.2	Locale (地域) 概念
857	
16.2.1	使用 Locale
858	
16.2.2	Locale Facet
864	

16.3 细究 Locale	866
16.4 细究 Facet	869
16.4.1 数值格式化 ( Numeric Formatting )	870
16.4.2 货币符号格式化 ( Monetary Formatting )	874
16.4.3 时间和日期格式化 ( Time and Date Formatting )	884
16.4.4 字符的分类和转换	891
16.4.5 字符串校勘 ( String Collation )	904
16.4.6 消息国际化 ( Internationalized Message )	905
17 数值	907
17.1 随机数及分布 ( Random Number and Distribution )	907
17.1.1 第一个例子	908
17.1.2 引擎 ( Engine )	912
17.1.3 细说引擎 ( Engine )	915
17.1.4 分布 ( Distribution )	917
17.1.5 细说分布 ( Distribution )	921
17.2 复数 ( Complex Number )	925
17.2.1 Class complex<>一般性质	925
17.2.2 Class complex<>运用实例	926
17.2.3 复数的各项操作	928
17.2.4 细说 Class complex<>	935
17.3 全局数值函数 ( Global Numeric Function )	941
17.4 Valarray	943
18 并发	945
18.1 高级接口 : async()和 Future	946
18.1.1 async()和 Future 的第一个用例	

946	
18.1.2 实例：等待两个 Task	
955	
18.1.3 Shared Future	
960	
18.2 低层接口：Thread 和 Promise	
964	
18.2.1 Class std::thread	
964	
18.2.2 Promise	
969	
18.2.3 Class packaged_task<>	
972	
18.3 细说启动线程（Starting a Thread）	
973	
18.3.1 细说async()	
974	
18.3.2 细说 Future	
975	
18.3.3 细说 Shared Future	
976	
18.3.4 细说 Class std::promise	
977	
18.3.5 细说 Class std::packaged_task	
977	
18.3.6 细说 Class std::thread	
979	
18.3.7 Namespace this_thread	
981	
18.4 线程同步化与 Concurrency（并发）问题	
982	
18.4.1 当心 Concurrency（并发）	
982	
18.4.2 Concurrent Data Access 为什么造成问题	
983	
18.4.3 什么情况下可能出错	
983	
18.4.4 解决问题所需要的性质（Feature）	
987	
18.5 Mutex 和 Lock	
989	
18.5.1 使用 Mutex 和 Lock	
989	
18.5.2 细说 Mutex 和 Lock	
998	
18.5.3 只调用一次	
1000	
18.6 Condition Variable（条件变量）	
1003	

18.6.1 Condition Variable (条件变量) 的意图	1003
18.6.2 Condition Variable (条件变量) 的第一个完整例子	1004
18.6.3 使用 Condition Variable (条件变量) 实现多线程 Queue	1006
18.6.4 细说 Condition Variable (条件变量)	1009
18.7 Atomic	1012
18.7.1 Atomic 用例	1012
18.7.2 细说 Atomic 及其高级接口	1016
18.7.3 Atomic 的 C-Style 接口	1019
18.7.4 Atomic 的低层接口	1019
19 分配器	1023
19.1 以应用程序开发者的角度使用 Allocator	1023
19.2 用户自定义的 Allocator	1024
19.3 以程序库开发者的角度使用 Allocator	1026
参考书目	1031
新闻组及论坛 (Newsgroup and Forum)	1031
书籍和网站	1032
索引	1037

# 《C++标准库（第2版）》

## 精彩短评

- 1、本书的第一句话：To those who care for people and mankind.
- 2、书很厚，结合了c++11的新特性进行讲解，看的很有滋味
- 3、绝对的与时俱进的好书，就是太厚了。
- 4、快速的看完了,觉得买的还是比较值得. 不过读过一遍以后基本就不会再翻了. 主要是 DASH 太好用了
- 5、本书刚出版不久，看了一下 比较全面 细致。是不可多得的好书。
- 6、只读了前六章，对标准库总体性的介绍的部分。一如既往的好。当作工具书用吧。
- 7、终于更新了C++11标准的第二版了啊 对于标准库的各种用法、注意事项及其原因介绍详尽 美中不足的是allocator的介绍还是太少了 不过也是因为这本书只是从使用者的角度来讲的吧
- 8、分类很科学，讲的很细，有代码实例，唯一美中不足就是type\_trait相关的东西一点都没讲，不知道为什么
- 9、排版不错，页数和英文版的一致。
- 10、这个书可以一直看下去。
- 11、内容，翻译，排版，都是绝佳啊！
- 12、侯老湿，您为啥不早点翻它？

# 《C++标准库（第2版）》

## 版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:[www.tushu111.com](http://www.tushu111.com)