

# Boost

## 程序库完全开发指南

### 深入C++“准”标准库

罗剑锋 著

第4版

# Boost

## 程序库完全开发指南

深入C++ “准” 标准库

—— (第4版) ——

罗剑锋 著

电子工业出版社

Publishing House of Electronics Industry

北京·BEIJING

## 内 容 简 介

Boost 是一个功能强大、构造精巧、跨平台、开源并且完全免费的 C++ 程序库，有着“C++ ‘准’标准库”的美誉。

Boost 由 C++ 标准委员会部分成员所设立的 Boost 社区开发并维护，使用了许多现代 C++ 编程技术，内容涵盖字符串处理、正则表达式、容器与数据结构、并发编程、函数式编程、泛型编程、设计模式实现等许多领域，极大地丰富了 C++ 的功能和表现力，能够使 C++ 软件开发更加简捷、优雅、灵活和高效。

本书基于 2017 年 4 月发布的 Boost1.64 版，介绍了其中的所有 140 余个库，并且结合 C++11/14/17 标准详细、深入地讲解了其中数十个库，同时实现了若干颇具实用价值的工具类和函数，可帮助读者迅速地理解、掌握 Boost 的用法并应用于实际的开发工作。

本书内容丰富、结构严谨、详略得当、讲解透彻，带领读者领略了 C++ 的最新前沿技术，相信会是每位 C++ 程序员的必备工具书。

未经许可，不得以任何方式复制或抄袭本书之部分或全部内容。  
版权所有，侵权必究。

### 图书在版编目 (CIP) 数据

Boost 程序库完全开发指南：深入 C++ “准”标准库 / 罗剑锋著. —4 版. —北京：电子工业出版社，2017.10  
ISBN 978-7-121-32460-4

I. ①B… II. ①罗… III. ①C++ 语言—程序设计—指南 IV. ①TP312.8-62

中国版本图书馆 CIP 数据核字 (2017) 第 195365 号

责任编辑：安 娜

印 刷：三河市鑫金马印装有限公司

装 订：三河市鑫金马印装有限公司

出版发行：电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本：787×980 1/16 印张：36.75 字数：812 千字

版 次：2017 年 10 月第 1 版

印 次：2017 年 10 月第 1 次印刷

印 数：2500 册 定价：99.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：(010) 88254888, 88258888。

质量投诉请发邮件至 [zltz@phei.com.cn](mailto:zltz@phei.com.cn)，盗版侵权举报请发邮件至 [dbqq@phei.com.cn](mailto:dbqq@phei.com.cn)。

本书咨询联系方式：010-51260888-819, [faq@phei.com.cn](mailto:faq@phei.com.cn)。

# 目录

第 0 章 导读	1	1.3.3 构建语言	13
0.1 关于本书	1	1.3.4 构建命令	14
0.2 读者对象	1	1.4 总结	14
0.3 术语与风格	2	第 2 章 时间与日期	15
0.4 语言标准	3	2.1 timer 库概述	15
0.5 本书的结构	4	2.2 timer	16
0.6 如何阅读本书	5	2.2.1 用法	16
0.7 本书的源码	5	2.2.2 类摘要	17
第 1 章 总论	7	2.2.3 使用建议	18
1.1 简介	7	2.3 progress_timer	18
1.1.1 获取方式	8	2.3.1 用法	18
1.1.2 目录结构	8	2.3.2 类摘要	19
1.1.3 使用方式	9	2.4 progress_display	19
1.2 开发环境	9	2.4.1 类摘要	20
1.2.1 操作系统和编译器	10	2.4.2 用法	21
1.2.2 快捷安装	10	2.4.3 注意事项	22
1.2.3 完全安装	10	2.5 date_time 库概述	23
1.2.4 定制安装	11	2.5.1 使用方式	23
1.2.5 编译验证	11	2.5.2 基本概念	24
1.3 构建工具	12	2.6 处理日期	24
1.3.1 安装方式	12	2.6.1 日期	25
1.3.2 构建脚本	12	2.6.2 创建日期对象	25

2.6.3	访问日期	27	3.2.3	用法	59
2.6.4	日期的输出	28	3.2.4	对比标准	61
2.6.5	转换 C 结构	29	3.3	shared_ptr	63
2.6.6	日期长度	29	3.3.1	类摘要	63
2.6.7	日期运算	30	3.3.2	操作函数	64
2.6.8	日期区间	31	3.3.3	用法	65
2.6.9	日期区间运算	33	3.3.4	工厂函数	67
2.6.10	日期迭代器	34	3.3.5	应用于标准容器	68
2.6.11	其他功能	35	3.3.6	应用于桥接模式	69
2.6.12	综合运用	36	3.3.7	应用于工厂模式	70
2.7	处理时间	38	3.3.8	定制删除器	72
2.7.1	时间长度	39	3.3.9	高级议题	73
2.7.2	操作时间长度	40	3.4	weak_ptr	76
2.7.3	时间精确度	42	3.4.1	类摘要	76
2.7.4	时间点	43	3.4.2	用法	77
2.7.5	创建时间点对象	44	3.4.3	对象自我管理	78
2.7.6	操作时间点对象	44	3.4.4	打破循环引用	79
2.7.7	转换 C 结构	45	3.5	intrusive_ptr	80
2.7.8	时间区间	46	3.5.1	类摘要	80
2.7.9	时间迭代器	46	3.5.2	用法	81
2.7.10	综合运用	47	3.5.3	引用计数器	82
2.8	date_time 库的高级议题	49	3.6	pool 库概述	83
2.8.1	编译配置宏	50	3.7	pool	83
2.8.2	自定义字面值	50	3.7.1	类摘要	84
2.8.3	格式化时间	51	3.7.2	操作函数	84
2.8.4	本地时间	51	3.7.3	用法	85
2.9	总结	53	3.8	object_pool	86
第 3 章	内存管理	55	3.8.1	类摘要	86
3.1	smart_ptr 库概述	55	3.8.2	操作函数	86
3.1.1	RAII 机制	55	3.8.3	用法	87
3.1.2	智能指针	56	3.8.4	更多的构造参数	88
3.2	scoped_ptr	57	3.9	singleton_pool	89
3.2.1	类摘要	57	3.9.1	类摘要	89
3.2.2	操作函数	58	3.9.2	用法	90
			3.10	pool_alloc	91

3.11 总结	91	4.6.5 相等与等价	121
<b>第4章 实用工具</b>	<b>93</b>	4.6.6 解引用操作符	122
4.1 noncopyable	93	4.6.7 下标操作符	123
4.1.1 原理	94	4.6.8 布尔转型操作符	124
4.1.2 用法	94	4.6.9 二元操作符	126
4.1.3 实现	95	<b>4.7 exception</b>	<b>126</b>
4.2 ignore_unused	96	4.7.1 标准库中的异常	127
4.2.1 基本用法	96	4.7.2 类摘要	127
4.2.2 模板用法	97	4.7.3 向异常传递信息	129
4.3 optional	97	4.7.4 错误信息类	130
4.3.1 类摘要	98	4.7.5 包装标准异常	132
4.3.2 操作函数	99	4.7.6 使用函数抛出异常	132
4.3.3 用法	100	4.7.7 获得更多信息	133
4.3.4 工厂函数	101	4.7.8 高级议题	134
4.4 assign	102	<b>4.8 uuid</b>	<b>136</b>
4.4.1 list_inserter	102	4.8.1 类摘要	136
4.4.2 operator+=	103	4.8.2 用法	137
4.4.3 operator ()	104	4.8.3 生成器	139
4.4.4 generic_list	105	4.8.4 增强用法	141
4.4.5 初始化容器	106	4.8.5 转换字符串	142
4.4.6 重复输入	108	4.8.6 摘要算法	143
4.4.7 操作非标准容器	109	<b>4.9 config</b>	<b>144</b>
4.4.8 其他议题	110	4.9.1 编译期字符串化	144
4.5 tribool	110	4.9.2 静态整型常量	145
4.5.1 类摘要	110	<b>4.10 utility</b>	<b>145</b>
4.5.2 用法	111	4.10.1 二进制常量	146
4.5.3 为第三态更名	112	4.10.2 调用函数名	147
4.5.4 输入/输出	113	<b>4.11 总结</b>	<b>148</b>
4.5.5 其他议题	113	<b>第5章 字符串与文本处理</b>	<b>151</b>
4.6 operators	114	5.1 lexical_cast	151
4.6.1 基本运算概念	116	5.1.1 函数声明	152
4.6.2 算术操作符	116	5.1.2 用法	152
4.6.3 基类链	118	5.1.3 错误处理	153
4.6.4 复合运算概念	119	5.1.4 转换对象的要求	154

5.1.5	应用于自定义类	155	5.6	总结	194
5.1.6	对比标准	156	第 6 章	正确性与测试	197
5.2	format	157	6.1	assert	197
5.2.1	简单的例子	157	6.1.1	基本用法	197
5.2.2	输入操作符	159	6.1.2	禁用断言	198
5.2.3	类摘要	160	6.1.3	扩展用法	199
5.2.4	格式化语法	161	6.2	static_assert	200
5.2.5	性能优化	162	6.2.1	定义	201
5.2.6	高级用法	162	6.2.2	用法	201
5.3	string_ref	164	6.2.3	使用建议	203
5.3.1	类摘要	164	6.3	lightweight_test	203
5.3.2	用法	166	6.3.1	测试断言	203
5.4	string_algo	167	6.3.2	用法	204
5.4.1	简单示例	168	6.3.3	测试元编程	205
5.4.2	算法概述	169	6.4	test	205
5.4.3	大小写转换	169	6.4.1	最小化测试	206
5.4.4	判断式(算法)	170	6.4.2	单元测试框架	207
5.4.5	判断式(函数对象)	172	6.4.3	测试断言	208
5.4.6	分类	172	6.4.4	测试主体	209
5.4.7	修剪	173	6.4.5	测试实例	210
5.4.8	查找	174	6.4.6	测试夹具	212
5.4.9	替换与删除	176	6.4.7	测试日志	214
5.4.10	分割	177	6.4.8	运行参数	215
5.4.11	合并	179	6.4.9	高级议题	216
5.4.12	查找(分割)迭代器	179	6.5	总结	219
5.5	xpressive	181	第 7 章	容器与数据结构	221
5.5.1	使用方式	181	7.1	array	221
5.5.2	正则表达式简介	182	7.1.1	类摘要	222
5.5.3	类摘要	183	7.1.2	操作函数	222
5.5.4	正则匹配	185	7.1.3	用法	223
5.5.5	正则查找	187	7.1.4	能力限制	224
5.5.6	正则替换	188	7.1.5	初始化	224
5.5.7	正则迭代	189	7.1.6	对比标准	225
5.5.8	正则分词	190	7.2	dynamic_bitset	225
5.5.9	高级议题	192			

7.2.1	类摘要	226	7.6.6	输入输出	261
7.2.2	创建与赋值	227	7.6.7	连结变量	262
7.2.3	容器操作	228	7.6.8	内部结构	263
7.2.4	基本运算	229	7.6.9	使用访问者模式	264
7.2.5	访问元素	230	7.6.10	高级议题	266
7.2.6	类型转换	231	7.7	<b>any</b>	269
7.2.7	集合操作	232	7.7.1	类摘要	269
7.2.8	综合运用	233	7.7.2	访问元素	270
7.3	<b>unordered</b>	234	7.7.3	用法	271
7.3.1	散列集合简介	234	7.7.4	简化操作	271
7.3.2	散列集合的用法	236	7.7.5	保存指针	272
7.3.3	散列映射简介	238	7.7.6	输出	273
7.3.4	散列映射的用法	239	7.7.7	高级议题	275
7.3.5	高级议题	241	7.8	<b>variant</b>	276
7.4	<b>bimap</b>	243	7.8.1	类摘要	276
7.4.1	类摘要	243	7.8.2	访问元素	277
7.4.2	基本用法	244	7.8.3	用法	278
7.4.3	值的集合类型	245	7.8.4	访问器	279
7.4.4	集合类型的用法	246	7.8.5	高级议题	281
7.4.5	使用标签类型	247	7.9	<b>multi_array</b>	283
7.4.6	使用 <code>assign</code> 库	249	7.9.1	类摘要	284
7.4.7	查找与替换	249	7.9.2	用法	285
7.4.8	投射	251	7.9.3	改变形状和大小	287
7.4.9	高级议题	252	7.9.4	创建子视图	287
7.5	<b>circular_buffer</b>	253	7.9.5	适配普通数组	289
7.5.1	类摘要	253	7.9.6	高级议题	290
7.5.2	用法	254	7.10	<b>property_tree</b>	292
7.5.3	环形结构	255	7.10.1	类摘要	293
7.5.4	空间优化	256	7.10.2	读取配置信息	294
7.6	<b>tuple</b>	257	7.10.3	写入配置信息	296
7.6.1	二元组	257	7.10.4	更多用法	297
7.6.2	类摘要	258	7.10.5	数据格式	298
7.6.3	创建与赋值	258	7.10.6	高级议题	300
7.6.4	访问元素	260	7.11	总结	302
7.6.5	比较操作	261			



第 8 章 算法	305	9.4 ratio	330
8.1 foreach	305	9.4.1 类摘要	330
8.1.1 用法	306	9.4.2 用法	331
8.1.2 详细解说	307	9.4.3 数字单位	331
8.1.3 更优雅的名字	308	9.4.4 字符串表示	333
8.1.4 支持的序列类型	308	9.5 crc	333
8.1.5 存在的问题	309	9.5.1 类摘要	334
8.2 minmax	310	9.5.2 预定义的实现类	334
8.2.1 用法	310	9.5.3 用法	335
8.2.2 存在的问题	311	9.6 random	336
8.3 minmax_element	311	9.6.1 随机数发生器	336
8.3.1 基本用法	312	9.6.2 随机数分布器	338
8.3.2 其他用法	312	9.6.3 变量发生器	343
8.4 algorithm	313	9.6.4 产生随机数据块	344
8.4.1 clamp	313	9.6.5 真随机数发生器	345
8.4.2 clamp_range	314	9.7 总结	348
8.4.3 hex/unhex	314	第 10 章 操作系统相关	349
8.5 总结	316	10.1 system	349
第 9 章 数学与数字	317	10.1.1 错误值	350
9.1 math.constants	317	10.1.2 错误类别	350
9.1.1 基本用法	318	10.1.3 错误代码	352
9.1.2 高级用法	318	10.1.4 错误异常	354
9.2 integer	319	10.2 chrono	354
9.2.1 整数特征	319	10.2.1 时间长度	355
9.2.2 标准整数类型	320	10.2.2 使用时间长度	356
9.2.3 整数类型模板类	322	10.2.3 时钟	358
9.3 rational	325	10.2.4 时间点	360
9.3.1 类摘要	326	10.2.5 综合运用	361
9.3.2 创建与赋值	327	10.3 cpu_timer	363
9.3.3 基本运算	327	10.3.1 时间类型	364
9.3.4 类型转换	328	10.3.2 cpu_timer	364
9.3.5 输入输出	328	10.3.3 auto_cpu_timer	366
9.3.6 分子与分母	328	10.3.4 定制输出格式	367
9.3.7 其他议题	329	10.4 filesystem	368

10.4.1	类摘要	368	11.2.1	工作原理	411
10.4.2	路径表示	370	11.2.2	绑定普通函数	413
10.4.3	可移植的文件名	372	11.2.3	绑定成员函数	414
10.4.4	路径处理	372	11.2.4	绑定成员变量	415
10.4.5	异常处理	375	11.2.5	绑定函数对象	416
10.4.6	文件状态	376	11.2.6	对比标准	416
10.4.7	文件属性	377	11.2.7	高级议题	417
10.4.8	文件操作	378	11.3	function	421
10.4.9	迭代目录	379	11.3.1	类摘要	421
10.4.10	实例 1	382	11.3.2	声明形式	422
10.4.11	实例 2	383	11.3.3	操作函数	423
10.4.12	实例 3	385	11.3.4	用法	424
10.4.13	文件流操作	386	11.3.5	使用 ref 库	425
10.5	program_options	387	11.3.6	用于回调	426
10.5.1	概述	387	11.3.7	对比 auto	428
10.5.2	选项值	389	11.3.8	对比 std::function	429
10.5.3	选项描述器	391	11.4	signals2	429
10.5.4	选项描述器的用法	392	11.4.1	类摘要	430
10.5.5	分析器	394	11.4.2	操作函数	431
10.5.6	存储器	395	11.4.3	用法	432
10.5.7	位置选项值	396	11.4.4	返回值	434
10.5.8	环境变量	398	11.4.5	合并器	434
10.5.9	分组选项信息	399	11.4.6	管理信号连接	436
10.5.10	高级用法	401	11.4.7	更灵活的管理信号连接	437
10.6	总结	404	11.4.8	自动管理连接	439
第 11 章	函数与回调	405	11.4.9	应用于观察者模式	441
11.1	ref	405	11.4.10	高级议题	444
11.1.1	类摘要	406	11.5	总结	449
11.1.2	基本用法	407	第 12 章	并发编程	451
11.1.3	工厂函数	407	12.1	atomic	451
11.1.4	操作包装	408	12.1.1	类摘要	452
11.1.5	综合应用	409	12.1.2	基本用法	454
11.1.6	对比标准	410	12.1.3	整数用法	455
11.2	bind	411	12.1.4	并发顺序一致性	456

12.2	thread	458	13.3	容器与数据结构	535
12.2.1	mutex	459	13.4	迭代器	536
12.2.2	lock_guard	461	13.5	函数对象与高级编程	536
12.2.3	unique_lock	463	13.6	泛型编程	538
12.2.4	lock_adapter	465	13.7	模板元编程	539
12.2.5	thread	467	13.8	预处理元编程	540
12.2.6	使用线程	469	13.9	并发编程	540
12.2.7	中断线程	472	13.10	数学与数字	541
12.2.8	thread_group	475	13.11	输入输出	542
12.2.9	call_once	476	13.12	操作系统相关	543
12.2.10	condition_variable	477	13.13	语言特性模拟	543
12.2.11	shared_mutex	481	13.14	杂项	544
12.2.12	future	483	13.15	总结	546
12.2.13	shared_future	486	第 14 章	设计模式	547
12.2.14	高级议题	487	14.1	创建型模式	548
12.3	asio	491	14.2	结构型模式	549
12.3.1	概述	491	14.3	行为模式	552
12.3.2	信号	497	14.4	其他模式	555
12.3.3	定时器	501	14.5	总结	557
12.3.4	网络通信	507	第 15 章	结束语	559
12.3.5	同步通信	514	15.1	未臻完美	559
12.3.6	异步通信	516	15.2	锦上添花	560
12.3.7	解析网络地址	521	15.3	工夫在诗外	563
12.3.8	使用协程	524	15.4	临别赠言	564
12.3.9	其他议题	526	附录 A	推荐书目	565
12.4	总结	530	附录 B	标准简述	567
第 13 章	组件速览	533	附录 C	关键字浅谈	571
13.1	算法	533			
13.2	字符串和文本处理	534			

# 第 0 章

## 导读

### 0.1 关于本书

C++是一种伟大的编程语言，某种程度上它甚至超越了编程语言的境界而升华为一种哲学。

C++又是一种多范式、可扩展的编程语言，支持多种编程风格——基于过程、基于对象、面向对象、函数式、泛型、模板元、自动机，非常自由灵活，易学难精，在不同编程风格间切换时必须小心谨慎以避免失误。

C++标准中出现的 STL（标准模板库）极大地改变了 C++ 程序员的编程思维，使“泛型”成为了 21 世纪以来程序开发界最流行的词汇之一。而 C++ 标准委员会成员所设立的 Boost 社区和开发的 Boost 程序库，更将“泛型”等现代 C++ 编程方法发挥到了极致。

Boost 程序库代表了目前 C++ 语言最新最前沿的技术，内容博大精深，丝毫不逊于经典的 STL，但同时也令很多人难以摸清门路，不得登堂入室而一窥究竟。有鉴于此，作者根据多年在实际开发中使用 Boost 库的经验并结合最新的 C++ 标准编写了本书，想为广大 C++ 程序员了解 C++ 的最新技术进展尽一份自己的力量。

本书的定位是“指南”（Guide、Introduction），而不是技术手册（Reference）或者使用说明（Manual），能够解答 90% 但不是所有 Boost 库相关的问题。但作者尽量做到让本书接近一本参考手册，使读者在阅读时能够脱离计算机，不至于频繁使用鼠标和键盘查询在线帮助或者源代码。

### 0.2 读者对象

本书适合有一定 C++ 基础的读者。

阅读本书需要有一定的 C++ 知识，基本要求是熟悉面向对象编程技术，如封装、继承、多态，此外还要理解名字空间、异常、模板<sup>①</sup>、泛型编程等高级特性（但不必非常深入），最好还能够了解设计模式和 C++ 标准库提供的容器、算法等各种组件。

如果读者是 C++ 初学者或还不具备以上所列的知识，建议先阅读附录 A 推荐书目里的技术书籍，然后再学习本书。如果手头刚好有推荐书目中的一本或两本，则可以一边翻阅这些书籍，一边学习本书。

总之，无论读者是 C++ 哪一层次的用户，现在或将来本书都会给您带来帮助。

### 0.3 术语与风格

本节列出了书中经常用到的专业术语和编程风格，以期与读者获得阅读的共识。

Boost 库并不是一个单一、平面化的程序库，而是有着复杂的内部结构，每个“库”可能是由其他许多更小的“库”组成的。因此，本书把程序库中所有组成部分统称为“组件”，“库”（Library）与“组件”（Component）这两个术语有时会通用。

namespace 这个术语有译作“命名空间”、“名称空间”、“名字空间”，本书称作“名字空间”。这只是作者个人习惯而已，如果在阅读过程中给读者造成了小小的困扰，还请谅解。

在使用 template 定义模板类或者模板函数时，本书统一使用 typename 而不是常见的 class，因为 typename 能够更清楚地向代码阅读者表明这是一个类型参数，而不一定是一个类（class）。但例外的是书中列出 Boost 源代码，会尽量保持其原始形式。

在命名函数或者类时，本书遵循 C++ 标准库和 Boost 的惯例，均采用小写形式，单词间以下画线分隔，如 demo\_class、rand\_bytes()，但并不要求读者也必须遵循这种命名方式，通常自己编写的类使用大写字母开头的单词命名会更好。

在 for 循环递增变量、指针或者迭代器对象的时候，本书统一使用 ++ 操作符的前置用法 (++i)，而不是后置用法 (i++)，因为前者不需要返回一个临时对象，执行效率更高。

“未定义行为”一词经常用来指代某些操作可能导致的不正确结果，如使用已失效的迭代

---

<sup>①</sup> 模板类和模板函数一般形式是 class\_or\_func<T>，例如 vector<int>。<> 在编程语言中通常被用于大小比较，因此其形式对于初学者很难接受，但这种形式具有良好的可读性，<> 可以被读作英文 of，比如 vector<int> 可以读作 vector of int，清晰地表明了类/函数和它的模板类型的关系，这样可以减少一部分对尖括号的不适应感，作者经常这么做，读者也可以试一下，很有效。

器、错误地使用指针，等等。对于“未定义行为”的一个较好（但不太精确）的定义是：程序在开发人员面前运行正常，在测试人员面前运行正常，但在老板或者最终用户面前运行时崩溃了。读者应当小心并尽量避免“未定义行为”，它是代码中的“定时炸弹”，如果它在调试的过程中爆炸了，那通常是最好的结果，因为它明确地告诉了我们代码存在问题。

## 0.4 语言标准

现在 C++ 有四个版本的国际标准：C++98、C++11、C++14 和 C++17<sup>①</sup>，本书主要使用 C++11 (ISO/IEC 14882:2011)，不含数字标识的“C++标准”一词通常就是指这个标准，但在涉及 C++ 某些语言特性时可能会明确标明具体的版本，有时还会以“C++11.x.y.z”的形式标明所引用 C++ 标准文档的章节号。

书中使用较多的新语言特性有四个：

- `nullptr` : 强类型的空指针；
- `auto/decltype` : 自动推导表达式类型；
- `for` : 基于范围的新式循环，形式更加优雅；
- `lambda` 表达式 : 又称闭包 (closure)，能够非常便捷地定义函数对象，通常的形式是 `[] (...){...}`，其中 `[]` 是捕获列表，`()` 是函数参数，`{}` 是函数体。

C++ 标准中定义的函数库本书称为“C++标准库”或者“STL”，但严格意义上 STL 与标准库并不等价，STL 只是标准库中的一个（很大的）子集，这么称呼有时候只是为了行文上的方便。

一般情况下使用 C++ 标准库都必须包含相应的头文件并且加上“`using namespace std`”语句，但标准库已经成为了 C++ 软件开发的基础设施，应用得非常频繁，因此书中的代码示例片断一般会将其略去。但有的情况下为了特别强调，会加上 `std` 名字空间前缀，如 `std::vector`。读者可以认为书中所有代码都默认包含了如下的头文件：

```
#include <string>           //标准字符串类
#include <iostream>        //标准输入输出
#include <vector>          //标准向量容器
#include <set>             //标准集合容器
#include <map>             //标准映射容器（关联数组）
#include <algorithm>      //标准算法
using namespace std;     //打开标准库的名字空间
```

<sup>①</sup> 实际上还应该有一个 C++03 标准，但因为它与 C++98 的差异很小，故本书忽略之。

为使读者对 C++ 标准能够有更多的了解，作者编写了一份简要介绍，作为本书附录供参考。

## 0.5 本书的结构

Boost 库组件繁多，相互关联也较多，如何排列其顺序是作者面临的一个颇为棘手的问题。

Boost 官方提供了两种基本方式：一种是按照组件的字母顺序，另一种是按照功能用途分类顺序，但这两种方式都不是组织本书结构的最佳手法。经反复考量，作者决定以难易度和实用程度对 Boost 库组件分类排序，采用由浅入深循序渐进的方式，先介绍较简单易用且实用程度高的库，然后逐步深入，介绍用法复杂的库，以期帮助读者尽快掌握 Boost 的使用方法。

对于每个 Boost 组件，本书通常首先简要介绍其功能，然后说明其头文件和编译方法（如果需要编译的话），列出类的声明概要，再使用例子讲解详细用法和注意事项，涉及其他 Boost 库组件时则以交叉引用的方式指明参考章节，最后是对该库的总结。

本书共分 16 章，各章的内容简介如下：

### ■ 第 1 章：总论

简要介绍 Boost 的历史、特点和获取方式，以及本书的开发环境和如何编译安装 Boost。

### ■ 第 2 章至第 13 章

第 2 章至第 13 章分门别类、由浅入深地介绍 Boost 库的各个组件，占据了本书的大部分篇幅，也是读者需要仔细阅读的内容。其中既包括如 timer、noncopyable 等简单的小工具，也包括 test、thread、asio 等用法复杂且功能强大的组件，Boost 1.64 版全部 140 余个库在本书中都可以找到阐述。

### ■ 第 14 章：设计模式

本章结合之前介绍的 Boost 库组件简要论述了推荐书目 [1] 中的 23 个设计模式和 4 个其他常用模式，以及 Boost 库使用这些设计模式的方法，从设计模式的抽象层次来加深理解 Boost 库。

### ■ 第 15 章：结束语

本章简单展望了 Boost 今后的发展，介绍其他可与 Boost 互为补充的开源 C/C++ 库，并对如何做一个好的程序员提出了自己的见解。

## ■ 附录

书末的附录也很有价值，列出了作者认为值得阅读的 C/C++ 经典书籍——它们也是作者编写本书时的案头必备参考资料，还有就是 C++ 标准简述和编程经验谈。

这些内容被放在附录部分并不是因为它们不重要，而仅仅是因为它们与全书的主题关联不大，但很值得阅读，读者也许会从中发现一些很有用的东西。

## 0.6 如何阅读本书

本书是一本介绍程序库的书籍，其中的很多组件是彼此独立的，所以在阅读完第 0 章（即本章）和第 1 章后，可以随意自由阅读其他章节。

读者既可以按照书的物理顺序循序渐进地逐页阅读，也可以查阅目录，然后直接跳到感兴趣的章节。不过对于大多数读者来说，作者还是推荐第一种方式，因为这可能会是学习效率更高、学习曲线更平滑的方式。

涉及编程语言，尤其是 C++，书中不可避免地会出现大量的代码片断，有的还可能很长，希望读者阅读时能有足够的耐心。这些代码都是精心编制的范例，将会指导读者如何使用 Boost 编写正确、高效的代码。

阅读时还需要注意一点：本书并不包含 Boost 库的所有接口说明，如果在阅读过程中对某些地方有疑虑，请参考 Boost 说明文档或者直接阅读 Boost 实现代码。

## 0.7 本书的源码

为了更好地方便读者利用本书学习研究 Boost 程序库，作者在 GitHub 网站上发布了书内所有示例程序的源代码，读者可以根据需要自取。<sup>①</sup>

本书源码的 GitHub 项目地址是：

```
https://github.com/chronolaw/boost\_guide.git
```

那么，欢迎来到 Boost 的世界。

---

<sup>①</sup> 有了 GitHub，再也不需要“附赠光盘”了，笑。



