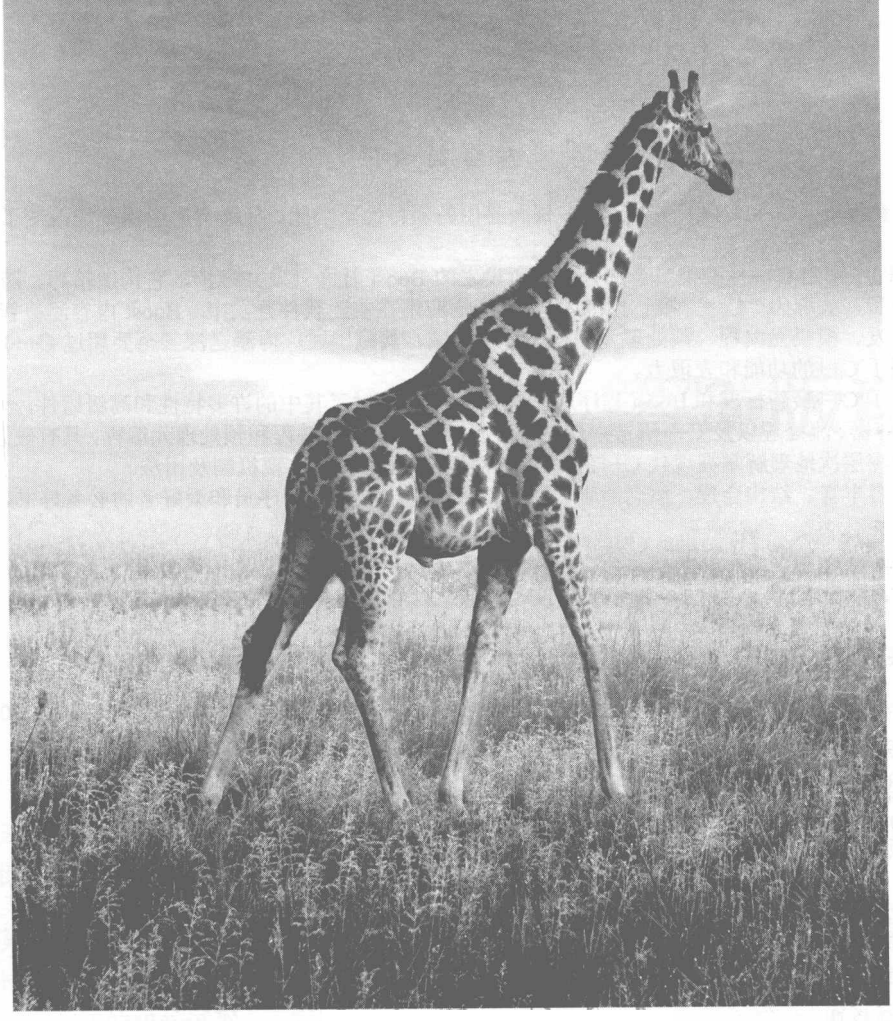

经典畅销书升级
深度解析C++11/14高级编程方法
C++11/14高级编程
Boost
程序库探秘（第3版）
罗剑锋◎著
清華大學出版社



C++11/14高级编程

Boost程序库探秘（第3版）

罗剑锋◎著

清華大學出版社
北京

内 容 简 介

C++的新标准（C++11/14）引入了许多强大易用的新特性新功能，从语言层面深刻地改变了C++的开发范式。

Boost程序库由C++标准委员会部分成员所设立的Boost社区开发并维护，它构造精巧、跨平台、开源并且完全免费，被称为“C++‘准’标准库”，已广泛应用在实际软件开发中。Boost内容涵盖智能指针、文本处理、并发、模板元编程、预处理元编程等许多领域，其范围之广内涵之深甚至要超过C++11/14标准，极大地增强了C++的功能和表现力。

本书基于C++最新标准和Boost程序库1.60版，深入探讨了其中的许多特性和高级组件，包括迭代器、函数对象、容器、流处理以及C++语言中最复杂最具威力的模板元编程和预处理元编程，具有较强的实用性，可帮助读者深层次地理解掌握现代C++的高级技术和Boost的内部实现机制及用法。

全书内容丰富、结构合理、概念清晰、讲解细致，是广大C++程序员和爱好者的必备好书。

图书在版编目（CIP）数据

C++11/14高级编程——Boost程序库探秘/罗剑锋著. —3版. —北京：清华大学出版社，2016

ISBN 978-7-302-44175-5

Ⅰ. ①C…　Ⅱ. ①罗…　Ⅲ. ①C语言－程序设计　Ⅳ. ①TP312

中国版本图书馆CIP数据核字（2016）第148590号

责任编辑： 袁金敏
封面设计： 刘新新
责任校对： 徐俊伟
责任印制： 宋　林

出版发行： 清华大学出版社
网　　址：http://www.tup.com.cn, http://www.wqbook.com
地　　址：北京清华大学学研大厦A座　　邮　　编：100084
社 总 机：010-62770175　　邮　　购：010-62786544
投稿与读者服务：010-62776969，c-service@tup.tsinghua.edu.cn
质量反馈：010-62772015，zhiliang@tup.tsinghua.edu.cn
印 装 者： 清华大学印刷厂
经　　销： 全国新华书店
开　　本： 185mm×235mm　　**印　张：** 31.75　　**字　数：** 796千字
版　　次： 2012年3月第1版　2016年9月第3版　　**印　次：** 2016年9月第1次印刷
印　　数： 1～3500
定　　价： 79.00元

产品编号：069426-01

目录

第 0 章　导读 …… 1

0.1　关于本书 …… 1
0.2　读者对象 …… 2
0.3　C++标准 …… 3
0.4　开发环境 …… 3
0.5　代码风格 …… 3
0.6　本书的结构 …… 4
0.7　如何阅读本书 …… 5
0.8　本书的源码 …… 6

第 1 章　全新的 C++语言 …… 7

1.1　概述 …… 8
1.2　左值与右值 …… 9
1.2.1　定义 …… 9
1.2.2　右值引用 …… 10
1.2.3　转移语义 …… 11
1.2.4　完美转发 …… 12
1.3　自动类型推导 …… 13
1.3.1　auto …… 13
1.3.2　decltype …… 15
1.3.3　decltype(auto) …… 17
1.4　面向过程编程 …… 17
1.4.1　空指针 …… 17
1.4.2　初始化 …… 18
1.4.3　新式 for 循环 …… 19
1.4.4　新式函数声明 …… 20
1.5　面向对象编程 …… 21
1.5.1　default …… 21
1.5.2　delete …… 22
1.5.3　override …… 23
1.5.4　final …… 24
1.5.5　成员初始化 …… 25
1.5.6　委托构造 …… 26
1.6　泛型编程 …… 27
1.6.1　类型别名 …… 27
1.6.2　编译期常量 …… 28
1.6.3　静态断言 …… 29
1.6.4　可变参数模板 …… 29
1.7　函数式编程 …… 31
1.7.1　lambda 表达式 …… 31
1.7.2　捕获外部变量 …… 32
1.7.3　类型转换 …… 34
1.7.4　泛型的 lambda 表达式 …… 35
1.8　并发编程 …… 35
1.9　面向安全编程 …… 37
1.9.1　无异常保证 …… 37
1.9.2　内联名字空间 …… 37
1.9.3　强类型枚举 …… 38
1.9.4　属性 …… 39
1.10　更多特性 …… 39
1.10.1　语言版本号 …… 39
1.10.2　超长整型 …… 40
1.10.3　原始字符串 …… 40
1.10.4　自定义字面值 …… 41
1.10.5　杂项 …… 43

1.11 总结 44

第 2 章 模板元编程简介 45

2.1 概述 45
2.2 语法元素 46
2.3 元数据 46
2.4 元函数 47
2.5 元函数转发 49
2.6 易用的工具宏 50
2.7 应用示例 51
2.8 总结 52

第 3 章 类型特征萃取 55

3.1 概述 55
3.2 元数据类别 56
3.2.1 基本类别 56
3.2.2 复合类别 58
3.3 元数据属性 60
3.3.1 基本属性 60
3.3.2 类相关属性 61
3.3.3 操作符重载属性 62
3.4 元数据关系 62
3.5 元数据运算 63
3.5.1 基本运算 63
3.5.2 特殊运算 65
3.6 解析函数元数据 67
3.7 实现原理 68
3.7.1 integral_constant 68
3.7.2 is_integral 69
3.8 应用示例 70
3.8.1 conditional 70
3.8.2 identity_type 71
3.8.3 declval 72
3.9 总结 73

第 4 章 实用工具 75

4.1 compressed_pair 75
4.1.1 空类 75
4.1.2 类摘要 77
4.1.3 构造与赋值 78
4.1.4 用法 78
4.1.5 实现原理 79
4.1.6 功能扩展 80
4.2 checked_delete 83
4.2.1 函数的用法 84
4.2.2 函数对象的用法 85
4.2.3 带检查的删除 87
4.2.4 实现原理 89
4.2.5 使用建议 90
4.3 addressof 90
4.3.1 用法 91
4.3.2 实现原理 92
4.3.3 使用建议 93
4.4 base_from_member 93
4.4.1 类摘要 93
4.4.2 用法 94
4.4.3 进一步的用法 96
4.5 conversion 98
4.5.1 标准转型操作符 98
4.5.2 多态对象的转型 99
4.5.3 polymorphic_downcast 101
4.5.4 polymorphic_cast 102
4.5.5 对引用转型 103
4.6 numeric conversion 104
4.6.1 bounds 104
4.6.2 numeric_cast 107
4.7 pointer 108
4.7.1 get_pointer 108
4.7.2 pointer_cast 109
4.7.3 pointee 110
4.7.4 indirect_reference 111
4.7.5 pointer_to_other 111
4.7.6 compare_pointees 113

4.7.7 pointer_traits ····· 114
4.8 总结 ····· 115

第 5 章 迭代器 ····· 117

5.1 概述 ····· 117
5.1.1 迭代器模式 ····· 117
5.1.2 标准迭代器 ····· 118
5.1.3 新式迭代器 ····· 119
5.1.4 标准迭代器工具 ····· 120
5.1.5 迭代器与算法 ····· 122
5.2 next_prior ····· 122
5.2.1 函数声明 ····· 123
5.2.2 用法 ····· 124
5.2.3 C++11/14 标准 ····· 125
5.3 iterator_traits ····· 125
5.3.1 标准迭代器特征类 ····· 126
5.3.2 类摘要 ····· 127
5.3.3 用法 ····· 127
5.4 iterator_facade ····· 128
5.4.1 迭代器的核心操作 ····· 128
5.4.2 类摘要 ····· 129
5.4.3 用法 ····· 131
5.5 iterator_adaptor ····· 135
5.5.1 类摘要 ····· 135
5.5.2 用法 ····· 136
5.6 迭代器工具 ····· 139
5.6.1 共享容器迭代器 ····· 139
5.6.2 发生器迭代器 ····· 141
5.6.3 逆向迭代器 ····· 143
5.6.4 间接迭代器 ····· 144
5.6.5 计数迭代器 ····· 145
5.6.6 函数输入迭代器 ····· 148
5.6.7 函数输出迭代器 ····· 151
5.6.8 过滤迭代器 ····· 153
5.6.9 转换迭代器 ····· 155
5.6.10 索引迭代器 ····· 157
5.6.11 组合迭代器 ····· 159
5.7 总结 ····· 161

第 6 章 区间 ····· 163

6.1 概述 ····· 163
6.2 特征元函数 ····· 164
6.3 操作函数 ····· 165
6.4 标准算法 ····· 166
6.4.1 返回原区间的算法 ····· 167
6.4.2 返回定制区间的算法 ····· 168
6.5 迭代器区间类 ····· 170
6.5.1 类摘要 ····· 170
6.5.2 用法 ····· 171
6.6 辅助工具 ····· 173
6.6.1 sub_range ····· 173
6.6.2 counting_range ····· 174
6.6.3 istream_range ····· 174
6.6.4 irange ····· 175
6.6.5 combined_range ····· 175
6.6.6 any_range ····· 176
6.7 适配器 ····· 178
6.7.1 适配器列表 ····· 178
6.7.2 用法 ····· 179
6.7.3 实现原理 ····· 180
6.8 其他议题 ····· 181
6.8.1 自定义区间类型 ····· 181
6.7.2 连接区间 ····· 182
6.9 总结 ····· 182

第 7 章 函数对象 ····· 185

7.1 hash ····· 185
7.1.1 类摘要 ····· 186
7.1.2 用法 ····· 186
7.1.3 实现原理 ····· 187
7.1.4 扩展 hash ····· 188
7.2 mem_fn ····· 191

7.2.1 工作原理 191
7.2.2 用法 192
7.2.3 其他议题 193
7.3 factory 194
7.3.1 类摘要 194
7.3.2 用法 195
7.3.3 value_factory 197
7.4 总结 197

第 8 章 指针容器 199

8.1 概述 199
8.1.1 入门示例 200
8.1.2 指针容器的优缺点 203
8.1.3 可克隆概念 204
8.1.4 克隆分配器 205
8.1.5 指针容器的分类 206
8.2 指针容器的共通功能 208
8.2.1 模板参数 208
8.2.2 构造与赋值 210
8.2.3 访问元素 211
8.2.4 其他功能 213
8.3 序列指针容器适配器 214
8.3.1 配置元函数 214
8.3.2 类摘要 215
8.3.3 接口解说 216
8.3.4 代码示例 216
8.4 ptr_vector 217
8.4.1 类摘要 218
8.4.2 用法 219
8.5 空指针处理 220
8.5.1 禁用空指针 220
8.5.2 使用空指针 220
8.5.3 空对象模式 221
8.6 关联指针容器的共通功能 223
8.6.1 类摘要 223
8.6.2 接口解说 224
8.7 集合指针容器适配器 225
8.7.1 配置元函数 225
8.7.2 ptr_set_adapter 226
8.8 ptr_set 227
8.8.1 类摘要 227
8.8.2 用法 228
8.9 ptr_unordered_set 228
8.9.1 类摘要 228
8.9.2 用法 229
8.10 映射指针容器适配器 230
8.10.1 配置元函数 230
8.10.2 ptr_map_adapter 231
8.11 ptr_map 233
8.11.1 类摘要 233
8.11.2 用法 234
8.12 ptr_unordered_map 234
8.12.1 类摘要 235
8.12.2 用法 235
8.13 使用 assign 库 236
8.13.1 向容器添加元素 236
8.13.2 初始化容器元素 237
8.14 使用算法 238
8.14.1 标准算法 238
8.14.2 序列指针容器的算法 242
8.14.3 关联指针容器的算法 244
8.15 其他议题 246
8.15.1 异常 247
8.15.2 间接函数对象 247
8.15.3 插入迭代器 248
8.15.4 使用视图分配器 248
8.15.5 可克隆性的再讨论 249
8.16 总结 250

第 9 章 侵入式容器 251

9.1 概述 251
9.1.1 手工实现链表 252

9.1.2 intrusive 库介绍 253
9.2 入门示例 254
9.2.1 使用基类挂钩 254
9.2.2 使用成员挂钩 255
9.3 基本概念 257
9.3.1 节点 257
9.3.2 节点特征 258
9.3.3 节点算法 258
9.3.4 值特征 260
9.3.5 挂钩 260
9.3.6 选项 262
9.3.7 处置器 263
9.3.8 克隆 264
9.4 链表 264
9.4.1 节点和算法 265
9.4.2 基类挂钩 266
9.4.3 成员挂钩 267
9.4.4 类摘要 267
9.4.5 基本用法 269
9.4.6 特有用法 271
9.5 有序集合 275
9.5.1 节点和算法 275
9.5.2 基类挂钩 276
9.5.3 成员挂钩 277
9.5.4 set 类摘要 277
9.5.5 基本用法 279
9.5.6 特有用法 280
9.6 无序集合 282
9.6.1 节点和算法 282
9.6.2 基类挂钩 283
9.6.3 成员挂钩 284
9.6.4 类摘要 284
9.6.5 基本用法 286
9.6.6 unordered_set 的特有用法 288
9.7 其他议题 290
9.7.1 链接模式 290
9.7.2 同时使用多个挂钩 291
9.7.3 万能挂钩 293
9.8 总结 293

第 10 章 多索引容器 295

10.1 概述 295
10.2 入门示例 296
10.2.1 简单的例子 296
10.2.2 复杂的例子 297
10.2.3 更复杂的例子 299
10.3 基本概念 302
10.3.1 索引 302
10.3.2 索引说明 303
10.3.3 键提取器 304
10.3.4 索引说明列表 304
10.3.5 索引标签 305
10.3.6 多索引容器 305
10.4 键提取器 306
10.4.1 定义 306
10.4.2 identity 307
10.4.3 member 308
10.4.4 const_mem_fun 310
10.4.5 mem_fun 311
10.4.6 global_fun 312
10.4.7 自定义键提取器 313
10.5 序列索引 313
10.5.1 索引说明 313
10.5.2 类摘要 314
10.5.3 用法 315
10.6 随机访问索引 317
10.6.1 索引说明 317
10.6.2 类摘要 317
10.6.3 用法 318
10.7 有序索引 320
10.7.1 索引说明 320
10.7.2 类摘要 320

10.7.3 基本用法 322
10.7.4 高级用法 323
10.8 散列索引 326
10.8.1 索引说明 326
10.8.2 类摘要 326
10.8.3 用法 327
10.9 修改元素 329
10.9.1 替换元素 329
10.9.2 修改元素 330
10.9.3 修改键 332
10.10 多索引容器 333
10.10.1 类摘要 333
10.10.2 用法 334
10.11 组合索引键 337
10.11.1 类摘要 337
10.11.2 用法 338
10.11.3 辅助工具 339
10.12 总结 341

第 11 章 流处理 343

11.1 概述 343
11.1.1 标准库的流处理 343
11.1.2 Boost 的流处理 345
11.2 入门示例 346
11.2.1 示例 1 346
11.2.2 示例 2 347
11.3 设备的特征 349
11.3.1 设备的字符类型 349
11.3.2 设备的模式 349
11.3.3 设备的分类 350
11.4 设备 351
11.4.1 概述 351
11.4.2 数组设备 352
11.4.3 标准容器设备 354
11.4.4 文件设备 355
11.4.5 空设备 357
11.5 过滤器 358
11.5.1 概述 358
11.5.2 设备链和管道 359
11.5.3 计数过滤器 361
11.5.4 换行过滤器 362
11.5.5 正则表达式过滤器（Ⅰ） 364
11.5.6 正则表达式过滤器（Ⅱ） 366
11.5.7 压缩过滤器 368
11.6 流 369
11.6.1 基本流 370
11.6.2 过滤流 371
11.7 流处理函数 373
11.8 定制设备 374
11.8.1 定制源设备 374
11.8.2 定制接收设备 377
11.9 定制过滤器 377
11.9.1 过滤器的实现原理 378
11.9.2 aggregate_filter 379
11.9.3 basic_line_filter 380
11.9.4 手工打造过滤器 381
11.10 组合设备 385
11.10.1 combine 385
11.10.2 compose 386
11.10.3 invert 387
11.10.4 restrict 389
11.10.5 tee 390
11.11 其他议题 391
11.11.1 对象的生存周期 391
11.11.2 与迭代器的比较 391
11.12 总结 392

第 12 章 泛型编程 395

12.1 enable_if 395
12.1.1 类摘要 396

12.1.2 应用于模板函数 397
12.1.3 应用于模板类 398
12.1.4 对比 C++11 标准 399
12.2 call_traits 399
12.2.1 类摘要 399
12.2.2 用法 400
12.2.3 实现原理 402
12.3 concept_check 403
12.3.1 概述 404
12.3.2 基本概念检查 405
12.3.3 函数对象概念检查 405
12.3.4 标准迭代器概念检查 406
12.3.5 新式迭代器概念检查 407
12.3.6 容器概念检查 409
12.3.7 区间概念检查 411
12.3.8 在函数声明中的概念检查 411
12.3.9 概念原型类 413
12.4 总结 414

第 13 章 模板元编程 415

13.1 概述 415
13.2 整数类型 416
13.2.1 简介 416
13.2.2 整数类型 418
13.2.3 bool 类型 419
13.2.4 基本运算 419
13.3 流程控制 421
13.3.1 if_和 if_c 421
13.3.2 eval_if 和 eval_if_c 422
13.4 容器 423
13.4.1 简介 424
13.4.2 vector 425
13.4.3 string 426
13.4.4 map 427
13.4.5 相关元函数 428
13.5 迭代器 429
13.5.1 简介 429
13.5.2 相关元函数 430
13.6 算法 431
13.6.1 插入器 431
13.6.2 查询算法 432
13.6.3 变换算法 433
13.6.4 运行时算法 434
13.7 高级用法 435
13.7.1 高阶元数据 436
13.7.2 占位符 437
13.7.3 bind 表达式 437
13.7.4 lambda 表达式 438
13.7.5 算法的高级应用 439
13.8 断言 441
13.8.1 基本断言 442
13.8.2 否定断言 442
13.8.3 关系断言 443
13.8.4 定制消息的断言 443
13.9 实例研究 444
13.9.1 泛型编程版本 444
13.9.2 元编程第 1 版 446
13.9.3 元编程第 2 版 449
13.10 总结 450

第 14 章 预处理元编程 453

14.1 概述 453
14.1.1 元数据 454
14.1.2 基本语法 454
14.1.3 特殊符号 456
14.1.4 特殊操作符 456
14.2 整数运算 457
14.3 常用元函数 458
14.3.1 ASSERT 458
14.3.2 IF 459

14.3.3 ENUM ····· 459
14.3.4 REPEAT ····· 460
14.4 高级数据结构 ····· 461
14.5 总结 ····· 462

第 15 章 现代 C++开发浅谈 ····· 463

15.1 基本原则 ····· 463
15.2 内存管理 ····· 467
15.3 容器、迭代器和算法 ····· 468
15.4 其他 ····· 469
15.5 结束语 ····· 471

附录 A 推荐书目 ····· 473

附录 B Boost 程序库组件索引 ····· 475

附录 C Boost 程序库安装简介 ····· 485

第0章 导读

0.1 关于本书

现在是 21 世纪的第二个十年，计算机编程语言领域已不再是早期几家独大的局面，而是风起云涌、各领风骚，新的语言不断出现，同时也有老的语言逐渐衰落。但从一些权威统计机构的数据来看，30 多年前诞生的 C++语言依然有着强大的生命力，稳稳保持着热门语言前三名的位置，即使是后来者 Java、C#、Python 等也未能撼动它的王者地位。

C++能够获得这样的成就绝非运气，而是源于它自身的优异品质。它兼容“中级语言”C，具有良好的结构和绝佳的运行效率，可以开发系统级软件；它又开创了许多新的现代编程范式，支持面向对象、泛型等技术，具有足够的抽象度，灵活方便，可以开发各种大型复杂的应用软件。在众多的编程语言中 C++可称得上是“全能选手”，可上可下，大至企业级应用，小至嵌入式系统，几乎没有什么事情是 C++做不到的。

多年以来 C++保持了良好的稳定性，是很多程序员学习和使用的首选，“对程序员最小限制”的哲学让我们不必受语言的约束，可以在计算机世界里任意驰骋。但 C++也没有故步自封，近几年新发布的 C++11/14 标准不仅全面继承传统，更带来了许多新的特性和强大易用的功能，例如自动类型推导、统一的初始化语法、内建的 lambda 表达式、可变参数模板、线程支持等，为 C++注入了新的活力，很大程度上改变了 C++的开发风格。

而在 C++标准之外，与标准委员会有密切联系的 Boost 程序库更进一步扩展了 C++的功能。

Boost 程序库充分利用了 C++的自扩展性这个最“神奇”的特性，在基本语言完全不变的情况下深入挖掘了语言的潜力，把泛型编程等高级技术发挥到了极致，开发出了上百个功能强大的库，涉及内存管理、文本处理、容器与数据结构、文件系统、并发、模板元编程、预处理元编程等许多领域。由于 Boost 是以社区的形式开发维护的，不受标准委员会的限制，版本迭代更快，比“保守”的官方标准更加“激进”，所以也被称为 C++标准的“试验场”，更有着“C++‘准’标准库”的美誉，范围之广内涵之深甚至要超过 C++11/14 标准，极大地扩展了 C++的功能。

C++11/14 和 Boost 代表了现代 C++的最新发展成果，在国外早已经是大行其道，并且在国内也逐渐流行了起来。以作者个人所知，国内一些软件公司都或多或少地应用了 C++11/14 和 Boost 库，也将能否掌握 C++11/14 和 Boost 作为评判个人能力的一个因素。但因为 C++11/14 和 Boost 库的博大精深远非一般的语言和开源库可比，很多程序员也只能使用其中的少量功能特性，不能完全发挥 C++11/14 和 Boost 的真正实力，还有为数不少的人出于偏见仍然把 C++11/14 和 Boost 视作畏途。①

为了能够让更多的朋友结识越来越美妙的 C++，笔者编写了本书，深入探究 C++11/14 和 Boost 的高级编程技术，希望读者借助本书能够汲取更多有用的知识，提升自己的能力，达到“知其然更知其所以然”的境界。

0.2 读者对象

本书定位于中高级读者，假设读者已经对 C++的语言特性有较深层次的理解，并且具备了一定的 C++知识。

在 C++11/14 中，面向对象的编程范式已经不是主要技术，更多的是使用泛型编程，所以本书的读者除了应熟悉基本的面向对象技术外，还应该对模板和泛型、模板的特化/偏特化、静态多态等 C++高级技术有所了解。

STL 是第一个真正把泛型编程技术表现的淋漓尽致的作品，是 C++标准库的核心，现代的很多 C++库都深受其设计思想和结构的影响，所以熟悉 STL 将非常有利于 C++11/14 和 Boost 程序库的学习。作为一个现代的 C++程序员，应该对 STL 的容器、迭代器和算法这三个最重要的部

① C++11/14 和 Boost 库目前的情形与十多年前的 STL 非常相似：当年 STL 甫出，国外的程序员欢欣鼓舞，而国内的程序员却是担心效率、开发风格等诸多问题，畏首畏尾。时至今日，STL 已经成为了 C++程序员的基本素质，曾经的责难都已经烟消云散，相信假以不长的时日 C++11/14 和 Boost 也会获得广大程序员的认同。

分熟稔于心（推荐书目[2] 对标准库有详尽的介绍，读者可参考）。

C++标准库和 Boost 程序库里的很多组件作者已经在推荐书目[3] 中做了较详细的阐述，本书中将会直接使用这些组件而少做或不做解释。如果读者对 Boost 所知不多，建议先阅读推荐书目[3] 然后再学习本书。

0.3 C++标准

C++的最新标准是 2014 年发布的 C++14，但目前很多编译器还未能完美支持，应用得也不够普及，故本书主要采用 C++11 标准。书中所称的“标准”“C++标准”通常指的是 C++11，而不是 C++98 或 C++14，更多的 C++标准知识请参考第 1 章。

涉及 C++标准文档时，本书会引用具体的章节号，形式是“C++11.章.节”，例如“C++11.20.2.3”表示标准的第 20 章第 20.2.3 节。

由于标准库已经成为了 C++的基础设施，故全书大部分代码均省略了标准库头文件和相应的“using namespace std”语句，请读者阅读时留意。不过个别情况下为了特别强调，偶尔会加上名字空间前缀，如 std::copy()。

0.4 开发环境

本书作者使用的开发环境是 Linux，具体如下所示。[①]

- 操作系统 ：Ubuntu 14.04（Linux 3.13.0）。
- 编译器 ：GCC 4.8.2（对 C++11 的支持较完善，但不支持 C++14）。
- Boost 程序库 ：1.60.0（2015 年 12 月发布）。

0.5 代码风格

遵循 C++标准和 Boost 的惯例，本书中的自定义类和函数均采用小写形式，模板参数列表

① 由于客观原因的限制，作者并未采用 Windows 和 VC 系列编译器，还请谅解。

统一使用更规范的 typename 而不是 class，递增操作使用效率更高的前置式（++i）。

为了改善代码的可读性，本书中的示例代码版式有一点小改进：某些需要强调的代码片段会用**粗体**或者*斜体*标明，读者可藉此迅速领会代码中的要点。

0.6 本书的结构

本书假设读者已经具备了一定的 C++和 Boost 知识，所以不再对 C++编译器和 Boost 库的安装和环境设置做介绍，而是直接切入主题讲解语言和库的使用。[①]

全书共 15 章，大致可以分为以下 8 个部分。

■ 第一部分，第 1 章：C++11/14 标准

主要介绍 C++11/14 的一些新特性，带领读者快速领略现代 C++开发风格和范式，以实用为目的，不求面面俱到和精确描述。

■ 第二部分，第 4～7 章：实用工具

本部分从 C++的深层次概念入手，介绍各种实用工具，涉及类型转换、对象的创建和删除、指针、迭代器、函数对象等诸多内容，学习它们可以对 C++的底层语言细节有更深入的了解，有助于构建更稳固健壮的程序。

■ 第三部分，第 8～10 章：新式容器

本部分较详细地阐述了三类新式容器：指针容器、侵入式容器和多索引容器，它们从不同的方向扩展了标准容器，功能强大内容庞杂，用起来有许多额外的考虑，读者会发现它们能够应用在许多特殊的场景。

■ 第四部分，第 11 章：流式处理

本部分论述了 C++提供的流处理功能。流处理是 C++诞生之初就有的功能，但多年来一直未被重视，boost.iostreams 在标准库的基础上构造了功能完备富有弹性的流式处理框架，可以灵活处理各种数据。

① Boost 程序库中的大部分组件都是以头文件的方式实现的，不需要编译，直接包含头文件即可。少数库需要编译才能使用，Boost 提供类似 make 的 b2 工具来完成库的编译和安装，可以参考附录 C 或者推荐书目[3] 。

■ 第五部分，第 2～3 章、第 12～13 章：泛型编程和模板元编程

本部分介绍 C++语言中最高级的技术：泛型编程和模板元编程，包括 type_traits、concept_check 和 mpl 等库，需要读者对 C++的泛型技术具有较深刻的理解。模板元编程虽然已经出现了很多年，但在国内仍然算是一个新的领域，相关的实践经验也不是很多，作者在这里也仅仅是做一个较为粗略的介绍。

■ 第六部分，第 14 章：预处理元编程

本部分介绍预处理元编程，它位于 C++语言体系之外，工作在编译之前的预处理阶段，使用宏的方式任意操作程序代码，不宜乱用，但在有些时候会很有用。

■ 第七部分，第 15 章：开发经验浅谈

本部分基于作者多年使用 C++的经历以条款的形式给出了一些开发经验，不一定完全正确，权做抛砖引玉之举，希望能够给读者起到一点借鉴的作用。

■ 第八部分，附录

书末的附录首先收录了作者认为值得阅读的 C++经典作品，然后是 Boost1.60 版全部组件的索引，最后是 Boost 库的简易安装手册。

0.7 如何阅读本书

对于所有读者来说，作者推荐首先阅读第 1 章和第 2 章，它们详细讲解了 C++11/14 的新特性和模板元编程，是全书的基础。第 1 章简要介绍了 C++新标准，学习难度不高，第 2 章虽然内容较深，但由于模板元编程在本书中的重要地位而被提到了全书的最开始，随后的许多章节都会用到其中的概念，熟悉模板元编程的基本概念对于理解本书是非常有帮助的。

接下来读者可以随个人意愿，或者按照书籍的物理顺序循序渐进逐页阅读，或者查阅目录，然后跳到感兴趣的章节。建议手里再拿一支笔，可以随时圈阅或标注，记录阅读时的领悟、疑问和心得，让本书成为您的学习笔记。

书中包含了大量作为示例的 C++源代码，都附加了详细的注释，希望读者能够耐心仔细阅读。另外本书还编制了交叉索引，阅读时涉及其他章节的内容都会指明具体位置，便于读者快速参考。

由于 C++和 Boost 库里很多组件的原理、用法较复杂，读者可结合附录的推荐书目阅读，最好再打开自己最熟悉的编辑器或集成开发环境，直接查看源代码加深理解。

0.8 本书的源码

为方便读者利用本书学习研究 C++11/14 和 Boost 程序库，作者在 GitHub 网站上发布了本书内所有示例程序的源码，地址是：

```
https://github.com/chronolaw/professional_boost.git
```