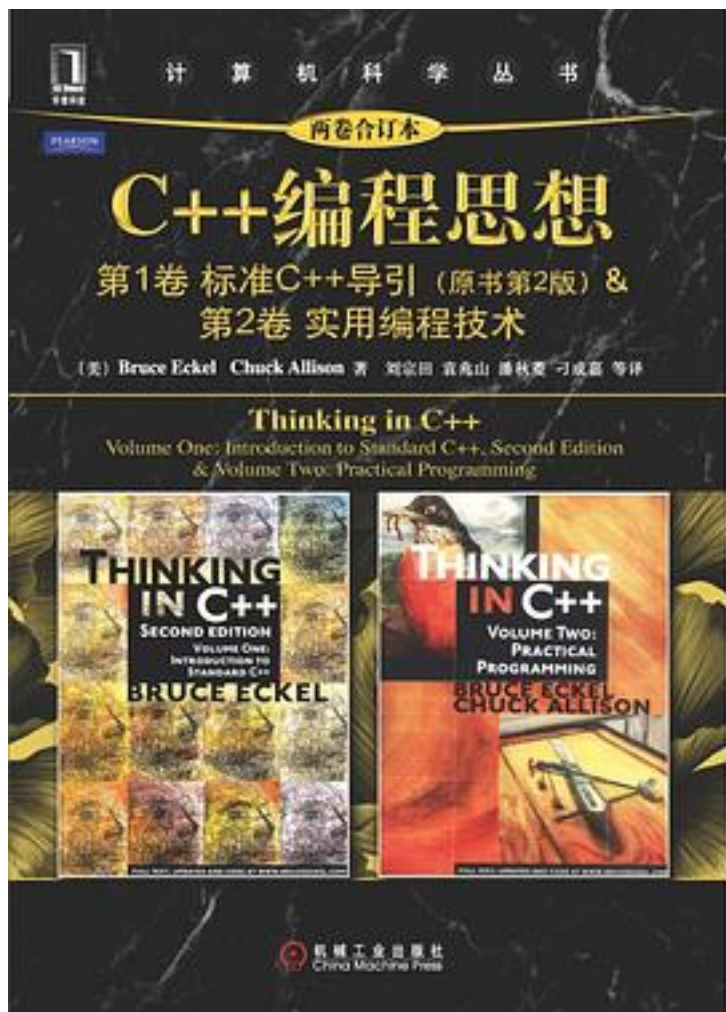


# C++编程思想（两卷合订本）



[C++编程思想（两卷合订本） 下载链接1](#)

著者: (美) Bruce Eckel

出版者: 机械工业出版社

出版时间: 2011-7

装帧: 平装

isbn: 9787111350217

本书是《C++编程思想》两卷的汇总。第1卷是在第1版的基础上进行了更加深入分析和

修改后的第2版，其内容、讲授方法、选用实例以及配套的练习别具特色，可以供不同程度的读者选择阅读。第2卷介绍了C++实用的编程技术和最佳的实践方法，深入探究了异常处理方法和异常安全设计；介绍C++的字符串、输入输出流的现代用法；解释多重继承问题的难点，描述了典型的设计模式及其实现，特别介绍了多线程处理编程技术。

在本书作者的个人网站[www.BruceEckel.com](http://www.BruceEckel.com)上提供：

本书的英文原文、源代码、练习解答指南、勘误表及补充材料。

本书相关内容的研讨和咨询。

本书第1卷及第2卷英文电子版的免费下载链接。

作者简介：

Bruce Eckel

是MindView公司的总裁，向客户提供软件咨询和培训。他是C++标准委员会拥有表决权的成员之一，他也是《Java编程思想》（该书第3版影印版及翻译版已由机械工业出版社引进出版）。他曾经写过另5本面向对象编程书籍，发表过150篇以上的文章，是多本计算机杂志的专栏作家。他经常参加世界各地的研讨会并进行演讲。

Chuck Allison 曾是《C/C++ Users》杂志的资深编辑，著有《C/C++ Code Capsules》一书。他是C++标准委员会的成员，犹他谷州立学院的计算机科学教授。他还是Fresh Sources公司的总裁，该公司专门从事软件培训和教学任务。

目录：出版者的话

出版说明

第1卷 标准C++导引

译者序 3

前言 5

第1章 对象导言 11

1.1 抽象的过程 11

1.2 对象有一个接口 12

1.3 实现的隐藏 14

1.4 实现的重用 15

1.5 继承：重用接口 15

1.5.1 is-a 关系和is-like-a 关系 18

1.6 具有多态性的可互换对象 18

1.7 创建和销毁对象 21

1.8 异常处理：应对错误 22

1.9 分析和设计 22

1.9.1 第0阶段：制定计划 24

1.9.2 第1阶段：我们在做什么 24

1.9.3 第2阶段：我们将如何建立对象 26

1.9.4 第3阶段：创建核心 28

1.9.5 第4阶段：迭代用例 29

1.9.6 第5阶段：进化 29

1.9.7 计划的回报 30

1.10 极限编程 30

1.10.1 先写测试 31

1.10.2 结对编程 32

1.11 为什么C++会成功	32
1.11.1 一个较好的C	32
1.11.2 延续式的学习过程	33
1.11.3 效率	33
1.11.4 系统更容易表达和理解	33
1.11.5 尽量使用库	33
1.11.6 利用模板的源代码重用	34
1.11.7 错误处理	34
1.11.8 大型程序设计	34
1.12 为向OOP转变而采取的策略	34
1.12.1 指导方针	35
1.12.2 管理的障碍	35
1.13 小结	37
第2章 对象的创建与使用	38
2.1 语言的翻译过程	38
2.1.1 解释器	38
2.1.2 编译器	39
2.1.3 编译过程	39
2.2 分段编译工具	40
2.2.1 声明与定义	40
2.2.2 连接	44
2.2.3 使用库文件	44
2.3 编写第一个C++程序	45
2.3.1 使用iostream类	45
2.3.2 名字空间	46
2.3.3 程序的基本结构	47
2.3.4 “Hello, World!”	47
2.3.5 运行编译器	48
2.4 关于输入输出流	48
2.4.1 字符数组的拼接	49
2.4.2 读取输入数据	49
2.4.3 调用其他程序	50
2.5 字符串简介	50
2.6 文件的读写	51
2.7 vector简介	52
2.8 小结	55
2.9 练习	56
第3章 C++中的C	57
3.1 创建函数	57
3.1.1 函数的返回值	58
3.1.2 使用C的函数库	59
3.1.3 通过库管理器创建自己的库	59
3.2 执行控制语句	60
3.2.1 真和假	60
3.2.2 if-else语句	60
3.2.3 while语句	61
3.2.4 do-while语句	61
3.2.5 for语句	62
3.2.6 关键字break 和 continue	63
3.2.7 switch语句	64
3.2.8 使用和滥用goto	65
3.2.9 递归	65
3.3 运算符简介	66
3.3.1 优先级	66

- 3.3.2 自增和自减 67
- 3.4 数据类型简介 67
  - 3.4.1 基本内建类型 67
  - 3.4.2 bool类型与true和false 68
  - 3.4.3 说明符 69
  - 3.4.4 指针简介 70
  - 3.4.5 修改外部对象 72
  - 3.4.6 C++引用简介 74
  - 3.4.7 用指针和引用作为修饰符 75
- 3.5 作用域 76
  - 3.5.1 实时定义变量 77
- 3.6 指定存储空间分配 78
  - 3.6.1 全局变量 78
  - 3.6.2 局部变量 79
  - 3.6.3 静态变量 80
  - 3.6.4 外部变量 81
  - 3.6.5 常量 82
  - 3.6.6 volatile变量 83
- 3.7 运算符及其使用 83
  - 3.7.1 赋值 83
  - 3.7.2 数学运算符 83
  - 3.7.3 关系运算符 85
  - 3.7.4 逻辑运算符 85
  - 3.7.5 位运算符 85
  - 3.7.6 移位运算符 86
  - 3.7.7 一元运算符 88
  - 3.7.8 三元运算符 88
  - 3.7.9 逗号运算符 89
  - 3.7.10 使用运算符时的常见问题 89
  - 3.7.11 转换运算符 90
  - 3.7.12 C++的显式转换 90
  - 3.7.13 sizeof—独立运算符 93
  - 3.7.14 asm 关键字 94
  - 3.7.15 显式运算符 94
- 3.8 创建复合类型 94
  - 3.8.1 用typedef命名别名 95
  - 3.8.2 用struct把变量结合在一起 95
  - 3.8.3 用enum提高程度清晰度 97
  - 3.8.4 用union节省内存 98
  - 3.8.5 数组 99
- 3.9 调试技巧 106
  - 3.9.1 调试标记 106
  - 3.9.2 把变量和表达式转换成字符串 108
  - 3.9.3 C语言assert()宏 108
- 3.10 函数地址 109
  - 3.10.1 定义函数指针 109
  - 3.10.2 复杂的声明和定义 109
  - 3.10.3 使用函数指针 110
  - 3.10.4 指向函数的指针数组 111
- 3.11 make：管理分段编译 111
  - 3.11.1 make的行为 112
  - 3.11.2 本书中的makefile 114
  - 3.11.3 makefile的一个例子 114
- 3.12 小结 116

- 3.13 练习 116
- 第4章 数据抽象 119
  - 4.1 一个袖珍C库 119
    - 4.1.1 动态存储分配 122
    - 4.1.2 有害的猜测 124
  - 4.2 哪儿出问题 125
  - 4.3 基本对象 126
  - 4.4 什么是对象 130
  - 4.5 抽象数据类型 131
  - 4.6 对象细节 131
  - 4.7 头文件形式 132
    - 4.7.1 头文件的重要性 132
    - 4.7.2 多次声明问题 133
    - 4.7.3 预处理器指示#define、#ifdef和#endif 134
    - 4.7.4 头文件的标准 134
    - 4.7.5 头文件中的名字空间 135
    - 4.7.6 在项目中使⤵用头文件 135
  - 4.8 嵌套结构 136
    - 4.8.1 全局作用域解析 138
  - 4.9 小结 139
  - 4.10 练习 139
- 第5章 隐藏实现 142
  - 5.1 设置限制 142
  - 5.2 C++的访问控制 142
    - 5.2.1 protected说明符 144
  - 5.3 友元 144
    - 5.3.1 嵌套友元 146
    - 5.3.2 它是纯面向对象的吗 148
  - 5.4 对象布局 148
  - 5.5 类 149
    - 5.5.1 用访问控制来修改Stash 151
    - 5.5.2 用访问控制来修改Stack 151
  - 5.6 句柄类 152
    - 5.6.1 隐藏实现 152
    - 5.6.2 减少重复编译 152
  - 5.7 小结 154
  - 5.8 练习 154
- 第6章 初始化与清除 156
  - 6.1 用构造函数确保初始化 156
  - 6.2 用析构函数确保清除 157
  - 6.3 清除定义块 159
    - 6.3.1 for循环 160
    - 6.3.2 内存分配 161
  - 6.4 带有构造函数和析构函数的Stash 162
  - 6.5 带有构造函数和析构函数的Stack 164
  - 6.6 聚合初始化 166
  - 6.7 默认构造函数 168
  - 6.8 小结 169
  - 6.9 练习 169
- 第7章 函数重载与默认参数 171
  - 7.1 名字修饰 172
    - 7.1.1 用返回值重载 172
    - 7.1.2 类型安全连接 172

- 7.2 重载的例子 173
- 7.3 联合 176
- 7.4 默认参数 178
  - 7.4.1 占位符参数 179
- 7.5 选择重载还是默认参数 180
- 7.6 小结 183
- 7.7 练习 183
- 第8章 常量 185
  - 8.1 值替代 185
    - 8.1.1 头文件里的const 186
    - 8.1.2 const的安全性 186
    - 8.1.3 聚合 187
    - 8.1.4 与C语言的区别 187
  - 8.2 指针 188
    - 8.2.1 指向const的指针 189
    - 8.2.2 const指针 189
    - 8.2.3 赋值和类型检查 190
  - 8.3 函数参数和返回值 191
    - 8.3.1 传递const值 191
    - 8.3.2 返回const值 191
    - 8.3.3 传递和返回地址 193
  - 8.4 类 195
    - 8.4.1 类里的const 196
    - 8.4.2 编译期间类里的常量 198
    - 8.4.3 const对象和成员函数 200
  - 8.5 volatile 204
  - 8.6 小结 205
  - 8.7 练习 205
- 第9章 内联函数 207
  - 9.1 预处理器的缺陷 207
    - 9.1.1 宏和访问 209
  - 9.2 内联函数 210
    - 9.2.1 类内部的内联函数 210
    - 9.2.2 访问函数 211
  - 9.3 带内联函数的Stash和Stack 215
  - 9.4 内联函数和编译器 218
    - 9.4.1 限制 219
    - 9.4.2 向前引用 219
    - 9.4.3 在构造函数和析构函数里隐藏行为 220
  - 9.5 减少混乱 220
  - 9.6 预处理器的更多特征 221
    - 9.6.1 标志粘贴 222
  - 9.7 改进的错误检查 222
  - 9.8 小结 225
  - 9.9 练习 225
- 第10章 名字控制 227
  - 10.1 来自C语言中的静态元素 227
    - 10.1.1 函数内部的静态变量 227
    - 10.1.2 控制连接 230
    - 10.1.3 其他存储类型说明符 232
  - 10.2 名字空间 232
    - 10.2.1 创建一个名字空间 232
    - 10.2.2 使用名字空间 234
    - 10.2.3 名字空间的使用 237

- 10.3 C++中的静态成员 238
  - 10.3.1 定义静态数据成员的存储 238
  - 10.3.2 嵌套类和局部类 241
  - 10.3.3 静态成员函数 242
- 10.4 静态初始化的相依性 244
  - 10.4.1 怎么办 245
- 10.5 替代连接说明 250
- 10.6 小结 250
- 10.7 练习 251
- 第11章 引用和拷贝构造函数 254
  - 11.1 C++中的指针 254
  - 11.2 C++中的引用 254
    - 11.2.1 函数中的引用 255
    - 11.2.2 参数传递准则 257
  - 11.3 拷贝构造函数 257
    - 11.3.1 按值传递和返回 257
    - 11.3.2 拷贝构造函数 261
    - 11.3.3 默认拷贝构造函数 265
    - 11.3.4 替代拷贝构造函数的方法 266
  - 11.4 指向成员的指针 267
    - 11.4.1 函数 269
  - 11.5 小结 271
  - 11.6 练习 271
- 第12章 运算符重载 274
  - 12.1 两个极端 274
  - 12.2 语法 274
  - 12.3 可重载的运算符 275
    - 12.3.1 一元运算符 276
    - 12.3.2 二元运算符 279
    - 12.3.3 参数和返回值 288
    - 12.3.4 不常用的运算符 290
    - 12.3.5 不能重载的运算符 295
  - 12.4 非成员运算符 296
    - 12.4.1 基本方针 297
  - 12.5 重载赋值符 297
    - 12.5.1 operator=的行为 298
  - 12.6 自动类型转换 306
    - 12.6.1 构造函数转换 306
    - 12.6.2 运算符转换 307
    - 12.6.3 类型转换例子 309
    - 12.6.4 自动类型转换的缺陷 310
  - 12.7 小结 312
  - 12.8 练习 312
- 第13章 动态对象创建 315
  - 13.1 对象创建 315
    - 13.1.1 C从堆中获取存储单元的方法 316
    - 13.1.2 operator new 317
    - 13.1.3 operator delete 317
    - 13.1.4 一个简单的例子 318
    - 13.1.5 内存管理的开销 318
  - 13.2 重新设计前面的例子 319
    - 13.2.1 使用delete void\*可能会出错 319
    - 13.2.2 对指针的清除责任 320
    - 13.2.3 指针的Stash 320

- 13.3 用于数组的new和delete 324
  - 13.3.1 使指针更像数组 325
- 13.4 耗尽内存 325
- 13.5 重载new和delete 326
  - 13.5.1 重载全局new和delete 327
  - 13.5.2 对于一个类重载new和delete 328
  - 13.5.3 为数组重载new和delete 330
  - 13.5.4 构造函数调用 332
  - 13.5.5 定位new和delete 333
- 13.6 小结 334
- 13.7 练习 334
- 第14章 继承和组合 336
  - 14.1 组合语法 336
  - 14.2 继承语法 337
  - 14.3 构造函数的初始化表达式表 339
    - 14.3.1 成员对象初始化 339
    - 14.3.2 在初始化表达式表中的内建类型 339
  - 14.4 组合和继承的联合 340
    - 14.4.1 构造函数和析构函数调用的次序 341
  - 14.5 名字隐藏 343
  - 14.6 非自动继承的函数 346
    - 14.6.1 继承和静态成员函数 349
  - 14.7 组合与继承的选择 349
    - 14.7.1 子类型设置 350
    - 14.7.2 私有继承 352
  - 14.8 protected 353
    - 14.8.1 protected继承 353
  - 14.9 运算符的重载与继承 353
  - 14.10 多重继承 355
  - 14.11 渐增式开发 355
  - 14.12 向上类型转换 356
    - 14.12.1 为什么要“向上类型转换” 357
    - 14.12.2 向上类型转换和拷贝构造函数 357
    - 14.12.3 组合与继承（再论） 359
    - 14.12.4 指针和引用的向上类型转换 360
    - 14.12.5 危机 360
  - 14.13 小结 361
  - 14.14 练习 361
- 第15章 多态性和虚函数 364
  - 15.1 C++程序员的演变 364
  - 15.2 向上类型转换 365
  - 15.3 问题 366
    - 15.3.1 函数调用捆绑 366
  - 15.4 虚函数 366
    - 15.4.1 扩展性 367
  - 15.5 C++如何实现晚捆绑 369
    - 15.5.1 存放类型信息 370
    - 15.5.2 虚函数功能图示 371
    - 15.5.3 撩开面纱 372
    - 15.5.4 安装vpointer 373
    - 15.5.5 对象是不同的 373
  - 15.6 为什么需要虚函数 374
  - 15.7 抽象基类和纯虚函数 375
    - 15.7.1 纯虚定义 378

- 15.8 继承和VTABLE 378
  - 15.8.1 对象切片 380
- 15.9 重载和重新定义 382
  - 15.9.1 变量返回类型 383
- 15.10 虚函数和构造函数 385
  - 15.10.1 构造函数调用次序 385
  - 15.10.2 虚函数在构造函数中的行为 386
- 15.11 析构函数和虚拟析构函数 386
  - 15.11.1 纯虚析构函数 388
  - 15.11.2 析构函数中的虚机制 389
  - 15.11.3 创建基于对象的继承 390
- 15.12 运算符重载 392
- 15.13 向下类型转换 394
- 15.14 小结 396
- 15.15 练习 397
- 第16章 模板介绍 400
  - 16.1 容器 400
    - 16.1.1 容器的需求 401
  - 16.2 模板综述 402
    - 16.2.1 模板方法 403
  - 16.3 模板语法 404
    - 16.3.1 非内联函数定义 405
    - 16.3.2 作为模板的IntStack 406
    - 16.3.3 模板中的常量 408
  - 16.4 作为模板的Stash和Stack 409
    - 16.4.1 模板化的指针Stash 411
  - 16.5 打开和关闭所有权 415
  - 16.6 以值存放对象 417
  - 16.7 迭代器简介 418
    - 16.7.1 带有迭代器的栈 425
    - 16.7.2 带有迭代器的PStash 427
  - 16.8 为什么使用迭代器 432
    - 16.8.1 函数模板 434
  - 16.9 小结 435
  - 16.10 练习 435
- 附录A 编码风格
- 附录B 编程准则
- 附录C 推荐读物
- 第2卷 实用编程技术
- 译者序 441
- 前言 442
- 第一部分 建立稳定的系统
- 第1章 异常处理 448
  - 1.1 传统的错误处理 448
  - 1.2 抛出异常 450
  - 1.3 捕获异常 451
    - 1.3.1 try块 451
    - 1.3.2 异常处理器 451
    - 1.3.3 终止和恢复 452
  - 1.4 异常匹配 453
    - 1.4.1 捕获所有异常 454
    - 1.4.2 重新抛出异常 454
    - 1.4.3 不捕获异常 455
  - 1.5 清理 456

- 1.5.1 资源管理 457
- 1.5.2 使所有事物都成为对象 458
- 1.5.3 auto\_ptr 460
- 1.5.4 函数级的try块 461
- 1.6 标准异常 462
- 1.7 异常规格说明 464
  - 1.7.1 更好的异常规格说明 467
  - 1.7.2 异常规格说明和继承 467
  - 1.7.3 什么时候不使用异常规格说明 468
- 1.8 异常安全 468
- 1.9 在编程中使用异常 471
  - 1.9.1 什么时候避免异常 471
  - 1.9.2 异常的典型应用 472
- 1.10 使用异常造成的开销 474
- 1.11 小结 476
- 1.12 练习 476
- 第2章 防御性编程 478
  - 2.1 断言 480
  - 2.2 一个简单的单元测试框架 482
    - 2.2.1 自动测试 483
    - 2.2.2 TestSuite框架 485
    - 2.2.3 测试套件 488
    - 2.2.4 测试框架的源代码 489
  - 2.3 调试技术 493
    - 2.3.1 用于代码跟踪的宏 494
    - 2.3.2 跟踪文件 494
    - 2.3.3 发现内存泄漏 495
  - 2.4 小结 499
  - 2.5 练习 500
- 第二部分 标准C++库
- 第3章 深入理解字符串 504
  - 3.1 字符串的内部是什么 504
  - 3.2 创建并初始化C++字符串 505
  - 3.3 对字符串进行操作 508
    - 3.3.1 追加、插入和连接字符串 508
    - 3.3.2 替换字符串中的字符 509
    - 3.3.3 使用非成员重载运算符连接 512
  - 3.4 字符串的查找 513
    - 3.4.1 反向查找 516
    - 3.4.2 查找一组字符第1次或最后一次出现的位置 517
    - 3.4.3 从字符串中删除字符 519
    - 3.4.4 字符串的比较 520
    - 3.4.5 字符串和字符的特性 523
  - 3.5 字符串的应用 527
  - 3.6 小结 531
  - 3.7 练习 531
- 第4章 输入输出流 534
  - 4.1 为什么引入输入输出流 534
  - 4.2 救助输入输出流 537
    - 4.2.1 插入符和提取符 537
    - 4.2.2 通常用法 540
    - 4.2.3 按行输入 541
  - 4.3 处理流错误 542
  - 4.4 文件输入输出流 544

- 4.4.1 一个文件处理的例子 544
- 4.4.2 打开模式 546
- 4.5 输入输出流缓冲 546
- 4.6 在输入输出流中定位 548
- 4.7 字符串输入输出流 550
  - 4.7.1 输入字符串流 551
  - 4.7.2 输出字符串流 552
- 4.8 输出流的格式化 555
  - 4.8.1 格式化标志 555
  - 4.8.2 格式化域 556
  - 4.8.3 宽度、填充和精度设置 557
  - 4.8.4 一个完整的例子 557
- 4.9 操纵算子 560
  - 4.9.1 带参数的操纵算子 560
  - 4.9.2 创建操纵算子 562
  - 4.9.3 效用算子 563
- 4.10 输入输出流程序举例 565
  - 4.10.1 维护类库的源代码 565
  - 4.10.2 检测编译器错误 568
  - 4.10.3 一个简单的数据记录器 570
- 4.11 国际化 573
  - 4.11.1 宽字符流 574
  - 4.11.2 区域性字符流 575
- 4.12 小结 577
- 4.13 练习 577
- 第5章 深入理解模板 580
  - 5.1 模板参数 580
    - 5.1.1 无类型模板参数 580
    - 5.1.2 默认模板参数 582
    - 5.1.3 模板类型的模板参数 583
    - 5.1.4 typename关键字 587
    - 5.1.5 以template关键字作为提示 588
    - 5.1.6 成员模板 589
  - 5.2 有关函数模板的几个问题 591
    - 5.2.1 函数模板参数的类型推断 591
    - 5.2.2 函数模板重载 594
    - 5.2.3 以一个已生成的函数模板地址作为参数 595
    - 5.2.4 将函数应用到STL序列容器中 598
    - 5.2.5 函数模板的半有序 600
  - 5.3 模板特化 601
    - 5.3.1 显式特化 601
    - 5.3.2 半特化 602
    - 5.3.3 一个实例 604
    - 5.3.4 防止模板代码膨胀 606
  - 5.4 名称查找问题 609
    - 5.4.1 模板中的名称 609
    - 5.4.2 模板和友元 613
  - 5.5 模板编程中的习语 617
    - 5.5.1 特征 617
    - 5.5.2 策略 621
    - 5.5.3 奇特的递归模板模式 623
  - 5.6 模板元编程 624
    - 5.6.1 编译时编程 625
    - 5.6.2 表达式模板 631

- 5.7 模板编译模型 636
  - 5.7.1 包含模型 636
  - 5.7.2 显式实例化 637
  - 5.7.3 分离模型 638
- 5.8 小结 639
- 5.9 练习 640
- 第6章 通用算法 642
  - 6.1 概述 642
    - 6.1.1 判定函数 644
    - 6.1.2 流迭代器 646
    - 6.1.3 算法复杂性 647
  - 6.2 函数对象 648
    - 6.2.1 函数对象的分类 649
    - 6.2.2 自动创建函数对象 649
    - 6.2.3 可调整的函数对象 652
    - 6.2.4 更多的函数对象例子 653
    - 6.2.5 函数指针适配器 658
    - 6.2.6 编写自己的函数对象适配器 662
  - 6.3 STL算法目录 665
    - 6.3.1 实例创建的支持工具 666
    - 6.3.2 填充和生成 669
    - 6.3.3 计数 670
    - 6.3.4 操作序列 671
    - 6.3.5 查找和替换 674
    - 6.3.6 比较范围 679
    - 6.3.7 删除元素 681
    - 6.3.8 对已排序的序列进行排序和运算 684
    - 6.3.9 堆运算 691
    - 6.3.10 对某一范围内的所有元素进行运算 691
    - 6.3.11 数值算法 697
    - 6.3.12 通用实用程序 699
  - 6.4 创建自己的STL风格算法 700
  - 6.5 小结 701
  - 6.6 练习 702
- 第7章 通用容器 706
  - 7.1 容器和迭代器 706
  - 7.2 概述 707
    - 7.2.1 字符串容器 711
    - 7.2.2 从STL容器继承 712
  - 7.3 更多迭代器 714
    - 7.3.1 可逆容器中的迭代器 715
    - 7.3.2 迭代器的种类 716
    - 7.3.3 预定义迭代器 717
  - 7.4 基本序列容器：vector、list和deque 721
    - 7.4.1 基本序列容器的操作 721
    - 7.4.2 向量 723
    - 7.4.3 双端队列 728
    - 7.4.4 序列容器间的转换 730
    - 7.4.5 被检查的随机访问 731
    - 7.4.6 链表 732
    - 7.4.7 交换序列 736
  - 7.5 集合 737
  - 7.6 堆栈 743
  - 7.7 队列 745

- 7.8 优先队列 748
- 7.9 持有二进制位 755
  - 7.9.1 bitset<n> 756
  - 7.9.2 vector<bool> 758
- 7.10 关联式容器 760
  - 7.10.1 用于关联式容器的发生器和填充器 763
  - 7.10.2 不可思议的映像 765
  - 7.10.3 多重映像和重复的关键字 766
  - 7.10.4 多重集合 768
- 7.11 将STL容器联合使用 771
- 7.12 清除容器的指针 773
- 7.13 创建自己的容器 774
- 7.14 对STL的扩充 776
- 7.15 非STL容器 777
- 7.16 小结 781
- 7.17 练习 781
- 第三部分 专题
- 第8章 运行时类型识别 785
  - 8.1 运行时类型转换 785
  - 8.2 typeid 操作符 789
    - 8.2.1 类型转换到中间层次类型 790
    - 8.2.2 void型指针 791
    - 8.2.3 运用带模板的RTTI 792
  - 8.3 多重继承 793
  - 8.4 合理使用RTTI 793
  - 8.5 RTTI的机制和开销 797
  - 8.6 小结 797
  - 8.7 练习 798
- 第9章 多重继承 800
  - 9.1 概论 800
  - 9.2 接口继承 801
  - 9.3 实现继承 803
  - 9.4 重复子对象 807
  - 9.5 虚基类 810
  - 9.6 名字查找问题 817
  - 9.7 避免使用多重继承 819
  - 9.8 扩充一个接口 820
  - 9.9 小结 823
  - 9.10 练习 823
- 第10章 设计模式 825
  - 10.1 模式的概念 825
  - 10.2 模式分类 826
  - 10.3 简化习语 827
    - 10.3.1 信使 827
    - 10.3.2 收集参数 828
  - 10.4 单件 829
  - 10.5 命令：选择操作 833
  - 10.6 消除对象耦合 836
    - 10.6.1 代理模式：作为其他对象的前端 837
    - 10.6.2 状态模式：改变对象的行为 838
  - 10.7 适配器模式 840
  - 10.8 模板方法模式 841
  - 10.9 策略模式：运行时选择算法 842
  - 10.10 职责链模式：尝试采用一系列

- 策略模式 843
- 10.11 工厂模式：封装对象的创建 845
  - 10.11.1 多态工厂 847
  - 10.11.2 抽象工厂 849
  - 10.11.3 虚构造函数 851
- 10.12 构建器模式：创建复杂对象 855
- 10.13 观察者模式 860
  - 10.13.1 “内部类”方法 862
  - 10.13.2 观察者模式举例 864
- 10.14 多重派遣 867
- 10.15 小结 873
- 10.16 练习 873
- 第11章 并发 875
  - 11.1 动机 875
  - 11.2 C++中的并发 876
  - 11.3 定义任务 878
  - 11.4 使用线程 879
    - 11.4.1 创建有响应的用户界面 880
    - 11.4.2 使用执行器简化工作 882
    - 11.4.3 让步 884
    - 11.4.4 休眠 885
    - 11.4.5 优先权 886
  - 11.5 共享有限资源 887
    - 11.5.1 保证对象的存在 887
    - 11.5.2 不恰当地访问资源 890
    - 11.5.3 访问控制 892
    - 11.5.4 使用保护简化编码 893
    - 11.5.5 线程本地存储 896
  - 11.6 终止任务 897
    - 11.6.1 防止输入/输出流冲突 897
    - 11.6.2 举例观赏植物园 898
    - 11.6.3 阻塞时终止 901
    - 11.6.4 中断 902
  - 11.7 线程间协作 906
    - 11.7.1 等待和信号 906
    - 11.7.2 生产者-消费者关系 909
    - 11.7.3 用队列解决线程处理的问题 912
    - 11.7.4 广播 916
  - 11.8 死锁 921
  - 11.9 小结 925
  - 11.10 练习 926
- 附录
  - 附录A 推荐读物
  - 附录B 其他
  - • • • • ([收起](#))

[C++编程思想（两卷合订本）\\_下载链接1](#)

标签

C++

编程

编程思想

C/C++

计算机

程序设计

计算机科学

C++

## 评论

翻译太差

-----  
大致翻了一遍。。

-----  
直接是 Bjarne Stroustrup 《C++程序设计语言》的讲解版

-----  
迅速刷一遍，很适合有C基础的人看。

-----  
有点无奈，去奋斗吧。

-----  
比较老的一本书，第一部主要是C语言转到C++语言，对于C++的原理讲的比较透彻，第二部主要讲C++的特性，相比现代C++，很多知识点都是老的。

-----  
大体知道模板那块咋回事了。

-----  
输入输出流、find  
copy函数之类我就没怎么看。感觉这些东西都是用到再查比较好，前半部分是国庆节期间看的，比较用心，后面的话自己也有点虎头蛇尾了。总体而言是一本好书，对我很有帮助。应该后面还会回来看的

-----  
看的是否是英文版并不重要，这书适合在一定强度的开发学习中同步使用，快速带过。很多时候搞不懂一个语言特性在干嘛或者设计的意图，主要是因为还没在实践中用过。不得不说，C++是一门需要学会退而求其次的语言，想要尽量学得高大全去炫技是毫无意义的。

-----  
感觉这本书属于中等层次，作为入门书的话开始有点不好切入，第一卷总体来说都是基础，第二卷有些深入，讲解有点繁杂，书挺厚，但是含金量不是很高，懂一些C++基础知识的同学可以把这本书读读

-----  
这本书可以当工具书，但是仔细研究感觉没啥必要，done了吧

-----  
翻译得跟坨屎一样

-----  
觉得翻译得一般，适合有C转向C++的人看呀~！

-----  
看完了第一卷，很棒的书

-----  
卷一对于学习C++的方方面面帮助很大，卷二的主要讲标志库及异常等方面的内容。或许是C++功力还不够，感觉对于C++学习只要看卷一就够了，卷二暂时还用不上。

-----  
入门经典

-----  
相比Primer更加有总结性，不是很散乱。第一卷尤其优秀，可以结合的内存模型一起看

-----  
另一本C++的书，也没有看完。

-----  
[C++编程思想（两卷合订本） 下载链接1](#)

## 书评

看的第三本c++的书，自然的描述、简单的单词、轻松的氛围，看完这本书，自己已经比较全面的了解C++语法、功能点，知道了不少C++还能做的事情。大师Bruce Eckel,佳作 Thinking in c++。温馨提示：一定要看英文版。

-----  
这本书带领我走入了C++和面向对象的大门，Bruce Eckel独特的思路可以让读者理清面向对象的脉络，抓住面向对象的核心，同时又能学习到C++语言本身最重要的那些知识。让我印象最为深刻的就是第一章就高屋建瓴地俯瞰了面向对象的核心思想。从抽象的演进讲到对象以及对象的接口， ...

-----  
虽然C++领域的经典书籍犹如过江之鲫、车载斗量，但其中的可供初学者入门的书籍却并不多。可能C++阵营里的牛人太多了，都不屑于写入门级新手教程。虽然Bjarne Stroustrup大叔说学习C++不需要学习C语言，但实践表明有C语言基础还是很有帮助的。另外大叔自己写的那本《The C++ Pro...

-----  
如果不考虑翻译得狗屁不通的话，这本书还是很经典的。当然虽然翻译得很差，但毕竟作者原文写得很好，也算抵消了一些翻译的硬伤。。。从内容上说，基本无可挑剔，唯一的遗憾是没有大的例子，毕竟要学OO，光看些玩具模型是不够的。

-----  
书大部分内容都比较深入。由于没有用过LINQ，所以关于LINQ的那章略过没有读。每个建议都是实践经验的终结，对于有一定C++使用经验的程序员能起到画龙点睛的作用。本书对于初学者不太适合，初学者可以去看Effective C++(第二版已经出版，但国内还没有引进，可以稍稍等等，第一...

-----  
都传说翻译差，看的时候就特别注意语意的流畅度，看了前面几章算是熟悉的内容，果然是差...倒是前言阅读尚可，囧。  
不过在国内这样的情况估计应该也可以理解，见多了慵懒的导师分配个任务，无聊的学生偷懒用个翻译软件完成个小作业罢了。如若如此，谨表示深深的鄙视。

-----  
对于计算机编程人员，就算截止到今天，能够很好地使用面向对象思想的人，也绝对不多，一是因为这个思想属于设计层面；另一个因为大家工作的层次都是面对函数，面对系统的机会很少。  
本书确实能让那些每天精通于函数的人们豁然开朗，发现原来，面向对象真的是可以利用的一种...

-----  
大三的时候，我刚学C++，被这本书深深的吸引，不得不说作者在教学上的造诣之高，远大于这本书所传授的知识本身，而在于一种写作的风格上面，确实是很多技术书籍可以借鉴的。  
大四的时候，我重温了一下，发现其实我自己还有好多东西遗漏或者遗忘，但是发现这本书相对于《C++从入...

-----  
传说中的C++变成死相~~不知道这本书怎么翻译出来的~~直接怀疑用金山快译出来稍微改一下就卖了~真的想学C++可以去看它的英文版或者C++ primer的中文版

-----  
如果是 C++ 新手, 或者, OO 新手, Thinking 系列第一卷绝对是一本好书.  
它给你解释了怎么从 C 到 C++ 的转变, 接着, 它慢慢的给你介绍 C++ 的各种东西, 先从数据抽象入手, 再到类, 再到各种特性, 如访问保护, 命名空间, 重载操作符, 继承, 多态, 模板(初级介绍)等等以及为什么...

-----  
书籍说明 最经典的C++书籍之一 适合在入门之后，在对整个C++的思维方式进行梳理  
大牛的经典作品，一定要读 当做进阶的C++书籍来读，一定会有收获的 阅读建议  
第三本C++书籍！！

-----  
E文原书读的头大, 还有,中文第一版的翻译,,,,实在不让人很满意 并且Bruce  
Eckel的书总带有那么一点点"禅意" 总体来说并不适合新手  
买一本放在书橱里收藏还是不错滴

-----  
友情提示：请直接读英文原版。2000  
年的书，内容有点老了，但基本思想是不变的，很适合入门，应该比下面的两本书都容  
易读。[C++ Primer 中文版（第5版）][C++ 程序设计语言（第1-3部分）（原书第4  
版）] 全书目录及各章节内容小结（不包括12和16两章，原因是Goo...

-----  
如果你想体验：每个字都认识但是每句话不读两遍愣是看不懂，看了几遍还是感觉云山  
雾绕。而且，大部分句子长到读到句末忘了开头说啥。那么，请参阅这本机翻无疑的经  
典巨著！  
不愧是参阅了《C++变成死相》第一卷的译者，在机翻水平方面和刘宗田不分伯仲，平  
分秋色！ 劝大家扔了， ...

-----  
只看了第一卷，觉得不好，也许我水平太渣，我觉得很不如c++  
primer，反正不喜欢这书，而且好多地方作者都是在说c。我没怎么接触过c，只在大一  
的c++教材里看过一些，而且我觉得c和c++不一样，所以看到c++书里说c比较反感，我  
觉得作者要是想说下与c的区别，可以用注明的形式啊。 ...

-----  
机械工业拿这么烂的东西出版吗？  
译者拿这么烂的翻译出来，是不是不吓死全国人民不罢休？  
原作者Bruce看了非吐血身亡不可。。。  
不是做开发的，就表碰开发类的书，理解得不透就表要写雷人害人的东西

-----  
没有看完这两卷的就不要评价这本书了，这本书可以说非常适合从c面向过程转c++面向  
对象的人看，虽然中文翻译某些地方比较差，但完全不影响你对那部分理解，因为作者  
每个讲解下面都有程序实例，对于程序员这是我们共同的语言，这些实例非常清晰，大  
爱本书里的例子，还有就是看完...

-----  
[大图] # C++编程思想 ## 常识 - impl惯用法 - 友元类 - RAII - 作用域 - 类 - 清晰 - const - 通常情况 符号表。没有地址 - 常量折叠 - 默认内部链接 - static const 与 enum hack - const\_cast<T\*> - volatile - 内联和宏 - 函数重载 extern C 命名空间 - 引用 - 指...

-----  
内容很详实，但是翻译的可是真的差啊，简直就是直接用机器翻译的，真的够差劲！关于内容没得说，该讲的都有讲解，而且具有一定的深度，非常好！但就是翻译影响了阅读体验啊！一句话要读好几遍才能理解翻译者的意思！唉！

-----  
首先，此书成书较早，对很多新的特性没有涉及，希望作者快出第三版:-D 我是看完《Effective C++》之后开始读的《Thinking in C++》，发现本书的文字比前者要易懂得多，当然也许是因为《Effective C++》的规模限制，作者不能展开讲解的缘故:-) 实际上两本书各有所长。《Thinkin...

-----  
[C++编程思想（两卷合订本） 下载链接1](#)