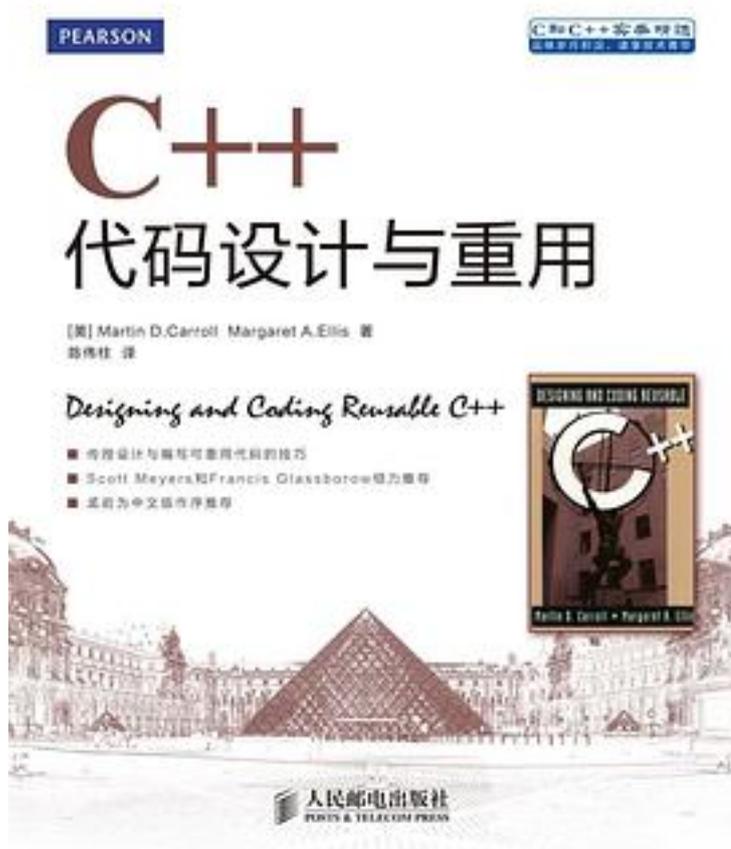


C++代码设计与重用



[C++代码设计与重用_下载链接1](#)

著者:Martin D.Carroll

出版者:人民邮电出版社

出版时间:2012-7

装帧:平装

isbn:9787115278289

"

本书全面展示如何使用C++编写可重用的代码，从而提高程序员的开发效率。

全书分为12章。包括重用性基本概念、类设计、扩展性、效率、错误、冲突、兼容性、继承、移植性、程序库等和重用相关的诸多话题。每一章的最后，通过总结和练习帮助你巩固概念、加深理解，参考文献和相关资料为你指明了深入学习的方向。

本书适合有一定C++经验的程序员阅读，也可供以提高代码重用性为专门学习方向的读者参考。"

作者介绍:

"Martin

Carroll是AT&T贝尔实验室的技术人员，他曾经用好几年的时间致力于设计和实现可重用的C++程序库，包括AT&T标准组件库（Standard Components Library）。他在Rutgers大学获得计算机科学博士学位。

Margaret Ellis 是The Annotated C++ Reference

Manual的合著者（另一个作者是大名鼎鼎的C++之父Bjarne Stroustrup），她主要致力于AT&T贝尔实验室、UNIX系统实验室和美国Novell公司的编译器开发。她曾获得加州大学计算机专业的硕士学位。

"

目录:"

目
录

第1章 重用性介绍 1

1.1 什么是重用性 1

1.1.1 提取代码来作为重用 2

1.1.2 可重用代码的基本特性 2

1.2 重用的神话 3

1.3 重用的障碍 4

1.3.1 非技术障碍 4

1.3.2 技术障碍 5

1.4 希望是否尚存 6

1.5 这本书能给我们带来什么 7

1.6 练习 8

1.7 参考文献和相关资料 9

第2章 类的设计 11

2.1 抽象性 11

2.2 正规函数 12

2.3 Nice类 14

2.4 存在最小标准接口吗 15

2.4.1 缺省构造函数 16

2.4.2 赋值运算符 17

2.4.3 拷贝构造函数 18

2.4.4 相等运算符 18

2.4.5 析构函数 18

2.5 浅拷贝和深拷贝 19

2.6 接口一致性 22

2.7 转型 25

2.7.1 多重所有权（Multiple Ownership） 26

2.7.2 敏感转型 26

2.7.3 不敏感转型 28

- 2.7.4 转型数目 (Fanout) 28
- 2.8 const关键字的使用 29
 - 2.8.1 抽象const对比位元const 29
 - 2.8.2 最大限度地使用const 31
 - 2.8.3 对const不安全的解释 32
- 2.9 总结 33
- 2.10 练习 34
- 2.11 参考文献和相关资料 37
- 第3章 扩展性 39
 - 3.1 扩展性的权衡 39
 - 3.2 扩展性和继承 40
 - 3.2.1 只继承基类的接口 41
 - 3.2.2 只继承基类的实现 42
 - 3.2.3 同时继承基类的接口和实现 43
 - 3.3 继承语义 (Semantie) 43
 - 3.4 继承的障碍 45
 - 3.4.1 非虚成员函数 45
 - 3.4.2 过度保护 47
 - 3.4.3 模块化不足 48
 - 3.4.4 friend关键字的使用 51
 - 3.4.5 成员变量过多 52
 - 3.4.6 非虚 (Nonvirtual) 派生 52
 - 3.4.7 妨碍继承的成员函数 53
 - 3.5 派生赋值问题 55
 - 3.6 允许入侵 (用户修改源代码) 继承 57
 - 3.7 总结 58
 - 3.8 练习 58
 - 3.9 参考文献和相关资料 60
- 第4章 效率 61
 - 4.1 效率和重用性 61
 - 4.2 程序创建时间 62
 - 4.2.1 编译时间 62
 - 4.2.2 实例化时间 64
 - 4.3 代码大小 69
 - 4.3.1 源文件分割 69
 - 4.3.2 外联的 (outlined) inline 71
 - 4.3.3 模板特化大小 71
 - 4.4 运行时间 72
 - 4.4.1 内联 (inlning) 72
 - 4.4.2 虚函数 74
 - 4.4.3 返回引用 76
 - 4.5 空闲存储空间 (free-store) 和堆栈空间 (stack space) 78
 - 4.5.1 使用高效的算法 79
 - 4.5.2 尽可能快地释放空闲资源 80
 - 4.5.3 静态对象 81
 - 4.5.4 庞大的对象 82
 - 4.6 效率的权衡 83
 - 4.6.1 实现更加困难 84
 - 4.6.2 使用更加困难 86
 - 4.7 总结 86
 - 4.8 练习 87
 - 4.9 参考文献和相关资料 89
- 第5章 错误 91
 - 5.1 可重用代码中的错误 91

- 5.2 错误检测 92
 - 5.2.1 函数前提条件 93
 - 5.2.2 表示不变性 93
- 5.3 处理错误 95
 - 5.3.1 程序库变量 95
 - 5.3.2 解决问题 95
 - 5.3.3 程序退出或者程序终止 (Exit or Abort) 96
 - 5.3.4 抛出异常 96
 - 5.3.5 返回错误值 97
 - 5.3.6 创建Nil值 98
 - 5.3.7 把无效的数据解释为有效的数据 99
 - 5.3.8 允许不确定的行为 99
- 5.4 资源限制 (Resource-Limit) 错误 100
 - 5.4.1 堆栈溢出 100
 - 5.4.2 用完空闲存储空间 101
 - 5.4.3 文件系统限制 102
- 5.5 异常安全性 103
 - 5.5.1 不一致的状态 104
 - 5.5.2 资源泄漏 105
- 5.6 总结 106
- 5.7 练习 107
- 5.8 参考文献和相关资料 110

第6章 冲突 111

- 6.1 全局名称 111
 - 6.1.1 翻译单元 112
 - 6.1.2 类的定义 112
 - 6.1.3 函数和数据的定义 114
 - 6.1.4 程序库的蕴涵意义 114
 - 6.1.5 命名约定 115
 - 6.1.6 namespace (名字空间) 结构 117
- 6.2 宏名称 118
 - 6.2.1 宏名称冲突 118
 - 6.2.2 去掉宏 119
 - 6.2.3 宏的命名约定 121
- 6.3 环境名称 121
- 6.4 Unclean程序库 122
- 6.5 Good-Citizen程序库 123
- 6.6 总结 123
- 6.7 练习 124
- 6.8 参考文献和相关资料 125

第7章 兼容性 127

- 7.1 向后和向前兼容性 127
- 7.2 兼容性的形式 128
- 7.3 理论源代码兼容性 129
- 7.4 实际源代码兼容性 130
- 7.5 链接兼容性 131
- 7.6 运行兼容性 133
- 7.7 进程兼容性 134
- 7.8 文档化不兼容性 135
- 7.9 非文档化特性 135
- 7.10 总结 136
- 7.11 练习 137
- 7.12 参考文献和相关资料 142

第8章 继承体系 143

- 8.1 根数目、深度和扇出数 143
- 8.2 体系类型 146
 - 8.2.1 直接体系 146
 - 8.2.2 接口体系 147
 - 8.2.3 对象工厂 (Object Factory) 149
 - 8.2.4 句柄体系 151
- 8.3 模板还是继承 154
 - 8.3.1 指针操纵 155
 - 8.3.2 派生要求 156
 - 8.3.3 实现不需要的函数 157
- 8.4 总结 158
- 8.5 练习 159
- 8.6 参考文献和相关资料 161
- 第9章 移植性 163
 - 9.1 有编写可移植代码的必要吗 163
 - 9.1.1 可移植性的优缺点 163
 - 9.1.2 目标代码和创建过程的可移植性 164
 - 9.2 不断发展的语言定义 165
 - 9.2.1 冲突 165
 - 9.2.2 实现的完整性 166
 - 9.3 不确定的行为 166
 - 9.3.1 排列方式和补全 (padding) 167
 - 9.3.2 地址操纵 168
 - 9.4 合法但不可移植的代码 169
 - 9.4.1 实现性定义的行为 169
 - 9.4.2 未经指定的行为 170
 - 9.5 实现依赖性 171
 - 9.6 可移植的数据文件 172
 - 9.7 模板实例化 173
 - 9.7.1 自动的实例化器 173
 - 9.7.2 人工实例化 177
 - 9.8 运行期程序库 179
 - 9.9 其他移植性问题 180
 - 9.10 总结 181
 - 9.11 练习 182
 - 9.12 参考文献和相关资料 184
- 第10章 使用其他程序库 185
 - 10.1 为何要重用其他程序库 185
 - 10.2 使用其他程序库的缺点 186
 - 10.2.1 获得可重用程序库 186
 - 10.2.2 效率 187
 - 10.2.3 冲突 187
 - 10.2.4 版本同步 188
 - 10.3 自含式 (Self-Contained) 程序库 190
 - 10.3.1 实现困难 190
 - 10.3.2 使用困难 191
 - 10.3.3 效率 192
 - 10.3.4 隔离 192
 - 10.4 总结 193
 - 10.5 练习 193
- 第11章 文档编制 195
 - 11.1 文档编制和重用性 195
 - 11.2 设计文档 196
 - 11.3 使用指南 196

11.3.1	对读者的背景知识了如指掌	197
11.3.2	用抽象的观点来编写	197
11.3.3	先解释普通用法	198
11.3.4	一次只解释一个事物	198
11.3.5	解释用法，不解释设计思路	199
11.3.6	简单清楚地编写	199
11.3.7	准确地使用语言	199
11.3.8	使用普遍接受的术语	200
11.3.9	深刻理解重载的术语	200
11.3.10	给出合法的、无错误的代码	201
11.3.11	保持简短的代码段	201
11.3.12	避免使用太大的函数	201
11.3.13	提供在线实例	202
11.4	参考手册	203
11.4.1	抽象化	203
11.4.2	语法接口	203
11.4.3	函数语义	205
11.4.4	模板参数约束	206
11.5	总结	207
11.6	练习	207
11.7	参考文献和相关资料	208
第12章	其他话题	209
12.1	静态初始化问题	209
12.1.1	构造和析构的时刻	210
12.1.2	程序库的蕴含意义	211
12.1.3	初始化函数	213
12.1.4	初始化检查	214
12.1.5	初始化对象	216
12.1.6	双构造	217
12.2	局部化开销原则	218
12.2.1	局部化开销和C++	219
12.2.2	局部化开销和程序库	219
12.3	内生类和外生类	220
12.4	迭代器	222
12.5	类耦合	224
12.6	推迟决定	226
12.7	总结	229
12.8	练习	229
12.9	参考文献和相关资料	232
	中英文术语对照表	233
	参考文献	261

"

• • • • • [\(收起\)](#)

[C++代码设计与重用_下载链接1](#)

标签

C++

计算机

C和C++实务精选

软件工程

C/C++

进阶

计算机科学

计算机技术

评论

成书较早，已经有很多精要的看法。适合进阶阅读。

近几年读过的最烂的C++书. 还好是部门图书馆买的...

老书（所以关于效率部分不可信了），讲解C++库设计与代码重用的各种坑……总评A，难度A+，推荐指数B

翻译的略糟糕，内容也太老了

大致读了一遍，感觉比预想的要差一点。

至少现阶段看不出这本书大部分篇幅的意义，一些有点意思的内容effectiveC++也有更好的解读

1995年的书，大的着眼点很有益处，但细节上已经和现代的C++脱离太多。不推荐此书。只看目录，并了解其着眼点即可。

后悔买了这本书。这是我现在看法，不代表以后也会持这种看法。

[C++代码设计与重用_下载链接1](#)

书评

此书的确成书较早，甚至完成于第一个C++标准勘定之前。买了放在书柜里面，也不知有多少年，随我辗转了多少地，今日才读完。前面的评论里有老师说，觉得书太老了后悔购买。我并不这么看。可能因为我水平较低的原因，觉得此书至少有几点是值得学习的：一是程序库设计的智慧。中...

不能说很差或是较差，但并不推荐。前几个星期买了，一直没有直接细翻，买的时候看了背面的两个推荐，一个是孟岩，一个是Scott Meyers（Effective C++系列的作者）。二牛推荐必属精品，于是没有细读，直接买下。这次去往返合肥，不想带什么行李，包里就只装了这本书，于是...

[C++代码设计与重用_下载链接1](#)