



C++

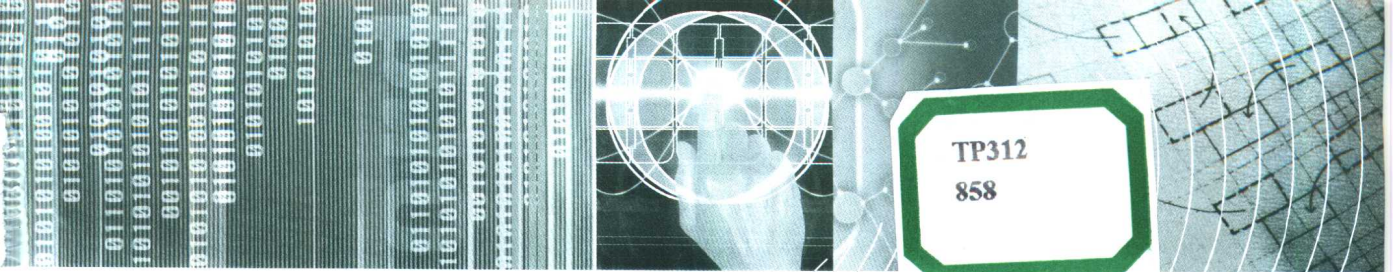
编码规范

C++ Code Standard

陈世忠 编著

摩托罗拉(中国)电子有限公司 审校

人民邮电出版社
POSTS & TELECOMMUNICATIONS PRESS



TP312
858

C++

编码规范

C++ Code Standard

陈世忠 编著

摩托罗拉(中国)电子有限公司 审校

北方工业大学图书馆



00516695

人民邮电出版社

图书在版编目(CIP)数据

C++编码规范/陈世忠编著. —北京: 人民邮电出版社, 2002.6
ISBN 7-115-10286-4

I. C... II. 陈... III. C 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字(2002)第 030370 号

C++编码规范

- ◆ 编 著 陈世忠
审 校 摩托罗拉(中国)电子有限公司
责任编辑 王文娟
- ◆ 人民邮电出版社出版发行 北京市崇文区夕照寺街 14 号
邮编 100061 电子函件 315@ptpress.com.cn
网址 <http://www.ptpress.com.cn>
读者热线 010-67180876
北京汉魂图文设计有限公司制作
北京顺义振华印刷厂印刷
新华书店总店北京发行所经销
- ◆ 开本: 800×1000 1/16
印张: 16.5
字数: 323 千字 2002 年 6 月第 1 版
印数: 1-5 000 册 2002 年 6 月北京第 1 次印刷

ISBN 7-115-10286-4/TP · 2861

定价: 32.00 元

本书如有印装质量问题, 请与本社联系 电话: (010) 67129223

内容提要

本书由 300 多条 C++ 编程原则组成，融合并提炼了许多人多年开发 C++ 程序积累下来的成熟经验，意在帮助读者形成良好的编程风格，迅速跨入业已存在的且具有相当高度的技术层次，并期望能够为提高代码的复用性提供积极的参考。

书中对每条原则给出了针对该原则的描述，并辅以实例加以说明，同时阐述了为什么要这么做的道理。每条原则相对独立又前后呼应，不需要花费大块的时间进行系统地阅读，按类细分的原则会让你学起来更轻松。

本书作者所在的部门达到 CMM 第 5 级（最高级）标准，书中列举的很多原则被列入该部门必须遵从的标准文档；且本书源于开发、用于开发，针对性比较强。本书适合有一定 C++ 语言基础的人阅读。如果你是一个初学者，希望一开始就养成良好的编程习惯，那么本书也会对你有所帮助。本书还可供软件开发项目组参考。

BJS42/01

前 言

为什么要写这本书

这本书最主要的思想来源是摩托罗拉公司 Steve Hawkes 和 Marc Peters 为寻呼系统部所写的 *Guidelines for Effective C++*，其次是 Scott Meyers 的两本著作 *Effective C++* 和 *More Effective C++*，另外，也参考了 SUN 公司的 *C++ User's Guide (Forte Developer 6 Upgrade 2)*、Erich Gamma 等的《设计模式——可复用面向对象软件的基础》、Bjarne Stroustrup 的 *The C++ Programming Language*，以及使用 Telelogic 公司 Logiscope 自动工具的一些经验。最后，还参考了许多不同组织、不同版本的关于 C++ 编码规范的相关文章和标准。

在本书中，我试图：

- 融合不同来源、不同观点。
- 借助别人的金玉良言和提示，把自己的经验增补进去。

这本书的素材是许多人多年开发 C++ 程序积累下来的经验，内容涉及 C++ 编程的许多方面，比如变量如何命名、类型如何使用、内存怎样管理、初始化需要注意的问题、封装 / 继承 / 重载 / 多态的使用指导、兼容性和性能的参考原则，甚至包括代码和注释如何排版才能有助于减少错误、增加可读性，以及文件和目录如何组织才能更有条理等。这些经验经过筛选、融合、提炼，最后分门别类呈献给大家，希望能从实践的角度出发，帮助大家更迅速、更全面、更深入地了解 and 掌握 C++ 编程以及面向对象技术，从而跨上业已存在的、坚实的且具有相当高度的技术层次，并能在此基础上探索我们未知的、更高层的技术问题。

另外需要强调的是，本书源于许多人的经验，而每个开发者也都有自己的编程体会，所以本书只是试图给大家一个参考，并没有强迫大家改变自己风格的意思。其实，经验只是表面的东西，真正有用的是其后隐藏的道理，也就是说，我们只要关心这里列举的每一条原则是否有道理、对理解问题是否有帮助就可以了。

本书的适用范围

一般来说,有关 C++ 的书籍大致可分为三类:第一类是入门级的,目的是让初学者掌握该语言。在这个层次,推荐《Turbo C++ 自学指南》,张玉亭等编译,是本古老的,不知原著,但三言两语之中,把概念讲得清晰透彻,深浅拿捏得恰如其分。

第二类是提高级的,目的是让阅读者抓住该语言的精髓,最终要达到挥洒自如的境界。Scott Meyers 著的 *Effective C++*、*More Effective C++* 是这个层次的经典之作。

第三类是开创级的,目的是点拨读者探索新的境界。在此级别,UML/SDL 语言(C++ 的终结者?)不可不知;一些有名的工具,如 Rational 的 Rose、Telelogic 的 Logiscope 等不可不知。Erich Gamma 等著的 *Design Patterns: Elements of Reusable Object-Oriented Software* 可纳入这个层次。

本书试图归于第二类,因而期望阅读者能有一定的 C++ 或面向对象程序设计的基础。

本书的组织

本书由 300 多条原则组成。基本上每一条原则都是相对独立的,所以可以按任何顺序阅读。根据它们所描述的领域,全书划分为 25 章。由于有的原则跨越多个领域,只好按自认为合适的方式划入某个章节。章节的顺序以及各章内原则的排列,希望能做到由小到大、从简入繁,并且有逻辑上的连贯性。但水平所限,难免有一定的随意性,请雅量海涵。另外,本书还试图围绕两个线索展开:一个是有关编码格式(风格)的,如第 1、17~23 等章节的大部分原则;另一个是有关用法的,如第 2~16 章的大部分原则。

感谢

在审校方面,我首先要感谢摩托罗拉公司的相关主管和法律部门,特别是我的老板阮祖望先生,他们的全面审查保证了本书符合摩托罗拉公司的利益和相关的法律法规。我还要感谢 C++ 方面的专家、我的同事和朋友潘杰先生,他逐字逐句的推敲使本书获益匪浅。我的另一位朋友陈瑛绮先生也对本书进行了大量审查,并提出许多宝贵的意见。除此以外,我还要感谢参加审校的王琛、贾壮和束玉文先生。

衷心感谢我的老板刘建山先生,他的全力支持使我得以在工作之余完成此书,并获得周围同事的技术援助。我还要感谢我的另一位老板 Mark Wickham 先生,他在法律方面给予我重要的指导。此外,向军先生和阮祖望先生的赞赏和鼓励也使我受到莫大的鼓舞。

最后，我要感谢我的家人，他们从来都是我最坚定、持久的支持者和最勤勉的督促者。

我的电子邮件地址为 shizhong@chen.com.cn，本书责任编辑的电子邮件地址为 wangwenjuan@ptpress.com.cn，欢迎与各位读者进行交流。

编者
2002 年 4 月

索引

I

inline 8, 41, 51, 123, 124, 125, 141,
149, 152, 165, 166, 167, 169, 204, 205,
206, 207

N

new/delete (关于 new 和 delete) 33,
39, 40, 49, 63, 67, 68, 71, 72, 73, 74,
97, 98, 112, 129, 142, 153, 190, 191,
192
NULL (关于 NULL) 7, 8, 28, 29, 33, 37,
38, 71, 72, 73, 94, 95, 100, 101, 214

V

void 7, 9, 11, 17, 19, 20, 21, 22, 25, 31,
32, 35, 36, 37, 38, 40, 42, 44, 48, 49,
50, 52, 58, 60, 61, 62, 63, 71, 72, 73,
82, 86, 87, 88, 89, 90, 93, 94, 95, 99,
105, 106, 107, 111, 113, 116, 127, 135,
138, 139, 149, 150, 153, 158, 163, 164,
186, 196, 197, 202, 204
volatile 104, 215

二画

二义性 52, 53, 93, 95, 107, 156, 180

三画

大写 / 小写 (关于大小写问题) 5, 6, 7,
8, 9, 12, 13, 18, 166, 196

四画

公共/public 1, 6, 9, 11, 12, 13, 15, 20,
21, 40, 42, 44, 45, 48, 50, 51, 52, 55,
56, 57, 58, 60, 61, 62, 63, 71, 72, 77,
79, 87, 90, 91, 93, 94, 95, 105, 106, 112,
128, 135, 136, 141, 150, 151, 152, 158,
162, 163, 166, 170, 172, 184, 186, 194,
195, 202

内存泄漏 39, 67, 68, 70, 132

内聚 (关于类的内聚性问题) 45, 46, 47

分支 64, 65, 105, 146, 205

切割 (关于对象的切割) 33

友元 41, 42, 43, 51, 52, 53, 98, 99, 101,
109, 110, 151, 152

引用 28, 29, 31, 32, 33, 38, 39, 40, 41,
43, 44, 47, 58, 62, 63, 69, 70, 78, 80,
89, 99, 112, 122, 128, 132, 134, 136,
148, 157, 162, 165, 166, 168, 169, 170,

171, 172, 173, 174, 175, 178, 184, 189,
195, 196, 201, 202, 205, 206, 207, 208,
209, 210, 213, 214, 215

五画

出口（关于函数出口） 36
可读性 19, 36, 118, 135, 165, 177, 209
头文件 31, 40, 87, 128, 162, 163, 165,
166, 167, 168, 169, 170, 171, 172, 173,
174, 175, 179, 184, 205, 206, 209, 214

六画

优化 40, 74, 146, 157, 185, 186, 191,
192, 199, 200, 201, 202, 203, 204, 206,
207, 209, 211, 212, 215
全局 10, 11, 12, 42, 51, 86, 87, 88, 92,
133, 194, 213
列表（初始化列表） 77, 78, 142
多态 / 虚（关于多态、虚函数、虚继承
等） 1, 2, 32, 33, 41, 42, 43, 44, 45,
48, 51, 56, 57, 58, 59, 60, 61, 62, 63,
64, 65, 66, 79, 83, 104, 105, 190, 192,
206, 208
多线程 / 锁（关于多线程问题） 50, 70,
91, 168, 169, 178, 179, 210, 213, 215
多重 / 菱形（关于多重继承，包括菱形
继承） 60, 65, 66, 81, 192, 205, 213
异常 26, 69, 70, 72, 105, 113, 129, 130,
131, 132, 133, 134, 169
有效、无效、失效 35, 38, 39, 40, 46, 47,
81, 109, 134, 183, 209, 216, 218
自定义 5, 32, 72, 73, 82, 84, 97, 98, 99,
100, 101, 105, 106, 112, 129, 131, 136,
154, 173, 187, 195, 208, 210

七画

初始化 28, 38, 68, 74, 75, 76, 77, 78,
79, 83, 85, 86, 87, 88, 118, 142, 178,
179, 194
返回（关于函数返回问题） 7, 8, 20, 28,
36, 37, 38, 39, 40, 52, 72, 76, 80, 90,
99, 101, 111, 140, 149, 183, 190, 192,
197, 201, 202, 206, 208, 210
宏 6, 9, 10, 12, 13, 28, 35, 121, 122,
123, 124, 125, 126, 127, 128, 172, 214
库 2, 12, 51, 166, 174, 185, 195, 196,
209, 210, 215
私有/private 12, 40, 45, 46, 48, 50,
53, 55, 56, 58, 60, 61, 71, 77, 87, 90,
112, 135, 151, 152, 184, 186
忘（避免遗忘） 36, 37, 74, 147, 157,
162, 207, 208, 209
位操作 / 移位（关于位操作、位类型
等） 27, 189, 199, 200

八画

参数（关于函数参数） 15, 21, 22, 28,
29, 31, 32, 33, 34, 35, 36, 39, 43, 51,
52, 55, 63, 64, 65, 67, 68, 69, 72, 74,
79, 80, 82, 83, 89, 91, 93, 94, 99, 101,
106, 107, 112, 123, 124, 125, 127, 131,
134, 149, 150, 151, 183, 184, 193, 194,
201, 202, 203, 205, 207, 210
命名冲突 13, 14, 54, 128, 213
定量（关于定量分析） 15, 26, 36, 47,
48, 59, 164, 199
审查（关于代码审查） 156, 216, 218
性能、效率、速度 3, 8, 26, 35, 41, 48,
50, 59, 62, 64, 75, 77, 85, 86, 105, 118,

121, 124, 136, 146, 177, 185, 186, 191,
192, 195, 199, 200, 201, 202, 204, 205,
206, 207, 208, 209, 210, 211, 212, 216,
218

构造(关于构造函数) 21, 28, 33, 43,
44, 53, 67, 68, 69, 70, 75, 76, 77, 78,
79, 80, 82, 83, 84, 87, 88, 106, 107, 112,
132, 152, 153, 158, 184, 200, 202, 203,
204, 205, 207, 208

析构(关于析构函数) 33, 62, 66, 67,
68, 69, 70, 75, 79, 82, 83, 84, 87, 88,
132, 153, 200, 202, 203, 204, 205, 207,
208

枚举 / enum(关于枚举) 5, 6, 9, 27, 28,
37, 47, 116, 122, 146, 151, 152, 183

注释 39, 40, 85, 111, 130, 131, 136, 146,
151, 155, 156, 157, 158, 159, 160, 161,
162, 163, 164, 167, 168, 169, 207

表达式 101, 115, 116, 123, 193, 205

受保护/protected 45, 48, 56, 60, 151,
152

九画

复用(关于代码复用) 1, 2, 3, 5, 13, 49,
89, 111, 129, 166, 168, 187

封装 1, 41, 42, 43, 45, 47, 60, 109, 157,
165, 187, 213

拷贝 21, 32, 33, 44, 53, 69, 70, 73, 79,
80, 81, 82, 83, 84, 85, 90, 134, 152,
166, 170, 183, 201, 207, 208, 210

指针 10, 12, 15, 26, 28, 29, 32, 33, 38,
39, 40, 41, 47, 62, 63, 64, 65, 69, 70,
71, 72, 73, 89, 91, 94, 95, 104, 112, 127,
132, 134, 148, 178, 184, 189, 190, 192,
194, 201, 202, 206, 207, 208, 209, 214

显式 25, 36, 45, 48, 53, 58, 60, 61, 76,
91, 103, 104, 105, 106, 136, 141, 151,
154, 183, 188, 189, 194, 202, 208

类型转换 25, 52, 53, 64, 103, 104, 105,
106, 107, 137, 184, 190, 202

绑定(关于动态、静态绑定) 63, 64, 206

重载 / overload(有关重载) 34, 51, 52,
62, 91, 93, 94, 95, 99, 107

顺序 77, 78, 86, 87, 98, 116, 146, 147,
148, 149, 151, 153, 167, 169, 173, 174,
193, 194, 195

十画

兼容 / 移植(关于兼容性、可移植性) 2,
27, 47, 95, 165, 166, 167, 169, 179,
186, 187, 188, 189, 190, 191, 192, 195,
196, 200, 205, 207

格式 5, 6, 12, 14, 72, 82, 99, 135, 144,
149, 160, 191, 195

浮点(关于浮点类型) 26, 116, 117, 190

继承 1, 41, 45, 47, 48, 55, 56, 57, 58,
59, 60, 61, 62, 63, 65, 66, 78, 81, 104,
109, 111, 178, 192, 206

缺省 37, 51, 53, 57, 58, 60, 61, 63, 64,
75, 78, 83, 93, 94, 97, 98, 101, 106, 112,
129, 131, 141, 207

十一画

常量 6, 9, 12, 13, 20, 27, 28, 41, 45, 47,
69, 77, 78, 79, 80, 82, 85, 89, 90, 91,
92, 99, 104, 121, 122, 148, 152, 154,
175, 184, 210

接口 1, 2, 33, 34, 41, 45, 47, 48, 49, 50,
57, 58, 60, 62, 63, 72, 84, 93, 98, 151,

152, 167, 171, 179, 214
 清除 75, 87, 88, 113, 134, 179, 215
 符号(关于符号类型) 25, 122, 137, 154,
 181, 188, 189, 191, 194, 200, 216
 维护 1, 13, 16, 27, 28, 31, 36, 40, 41,
 47, 48, 65, 94, 100, 115, 119, 135, 148,
 155, 157, 162, 172, 177, 179, 180, 210,
 214
 缀(关于前缀、后缀) 6, 10, 11, 12, 13,
 15, 20, 21, 99, 100, 128, 150, 165, 166,
 167, 172, 204
 隐式 25, 32, 33, 34, 52, 53, 75, 77, 78,
 79, 80, 90, 103, 105, 106, 107, 184,
 188, 202, 204, 205, 206, 210

十二画

嵌套 36, 54, 87, 113, 114, 136, 159, 160,
 171
 强迫、强制、迫使 31, 43, 45, 57, 64, 84,
 89, 92, 103, 104, 137, 158, 189
 循环 85, 86, 118, 119, 142, 144, 162,
 169, 185, 200, 205
 编译 2, 10, 28, 31, 32, 33, 34, 37, 43,
 48, 52, 53, 54, 56, 61, 63, 64, 68, 74,
 78, 79, 80, 83, 84, 85, 86, 89, 95, 98,
 99, 101, 103, 104, 105, 106, 112, 113,
 114, 116, 121, 123, 124, 125, 128, 130,
 146, 148, 151, 154, 159, 165, 166, 167,
 168, 170, 171, 172, 173, 174, 177, 178,
 179, 180, 181, 183, 184, 185, 186, 187,
 188, 189, 190, 191, 192, 193, 194, 196,
 197, 199, 200, 201, 202, 203, 204, 205,

206, 207, 208, 209, 212, 213, 215, 216,
 218
 赋值 21, 53, 69, 70, 75, 79, 80, 81, 82,
 83, 84, 98, 99, 115, 116, 152, 181, 183,
 201, 203, 207, 208

十三画

溢(关于溢出问题) 26, 49, 190, 192,
 193, 204
 禁止 40, 48, 53, 106, 151, 152, 183,
 202, 208, 211, 213

十四画

模板/template 6, 9, 23, 56, 57, 95,
 103, 111, 112, 113, 114, 124, 131, 141,
 149, 150, 163, 164, 167, 168, 185, 186
 静态(关于静态成员等) 10, 11, 47, 48,
 63, 64, 78, 79, 86, 87, 88, 91, 92, 153,
 154, 194, 206, 208

十五画

耦合(关于类间的耦合度问题) 45, 47,
 60, 109, 179

十六画

操作符 34, 52, 68, 72, 73, 75, 97, 98,
 99, 100, 107, 109, 114, 136, 153, 204

目 录

第 0 章 为什么要用 C++	1
0.1 原因	1
0.2 语言的发展和代码复用	2
0.3 代码复用的特点	2
0.4 代码复用对我们的影响	3
第 1 章 命名原则	5
原则 1.1 关于类型名	5
原则 1.2 关于变量和函数名	6
原则 1.3 关于全大写的函数名（建议）	7
原则 1.4 关于宏、常量和模板名	9
原则 1.5 关于指针标识符名（供参考）	10
原则 1.6 关于变量名前缀（供参考）	10
原则 1.7 关于匿名命名空间级标识符的前缀	12
原则 1.8 减少匿名命名空间级标识符	13
原则 1.9 命名时避免使用国际组织占用的格式	14
原则 1.10 名字要本着清楚、简单的原则	14
原则 1.11 尽量用可发音的名字	15
原则 1.12 尽量用英文命名	16
原则 1.13 尽量选择通用词汇并贯穿始终	16
原则 1.14 避免用模棱两可、晦涩或不标准的缩写	16
原则 1.15 避免使用会引起误解的词汇	17
原则 1.16 减少名字中的冗余信息	17
原则 1.17 建议起名尽量通俗，太专一会限制以后的扩展	17
原则 1.18 名字最好尽可能精确地表达其内容	18
原则 1.19 避免名字中出现形状混淆的字母或数字	18

原则 1.20	命名类和成员使得“object.method()”有意义	18
原则 1.21	类和对象名应是名词	19
原则 1.22	实现行为的类成员函数名应是动词	19
原则 1.23	类的存取和查询成员函数名应是名词或形容词	20
原则 1.24	变量名应是名词	20
原则 1.25	布尔型的名字要直观	21
原则 1.26	关于函数的左值参数和右值参数名	21
原则 1.27	避免局部名和外层的名字冲突	22
原则 1.28	用 a、an、any 区分重名（参数）	22
原则 1.29	模板类型名应有意义	23
 第 2 章 类型的使用		25
原则 2.1	避免隐式声明类型	25
原则 2.2	慎用无符号类型	25
原则 2.3	少用浮点数除非必须	26
原则 2.4	用 typedef 简化程序中的复杂语法	26
原则 2.5	少用 union	27
原则 2.6	慎用位操作	27
原则 2.7	用 enum 取代（一组相关的）常量	27
原则 2.8	使用内置 bool 类型	28
原则 2.9	（尽量）用引用取代指针	28
 第 3 章 函数		31
原则 3.1	一定要做到先定义后使用	31
原则 3.2	函数原型声明放在一个头文件中	31
原则 3.3	函数无参数一定要用 void 标注	31
原则 3.4	对于内置类型参数应传值（除非函数内部要对其修改）	32
原则 3.5	对于非内置类型参数应传递引用（首选）或指针	32
原则 3.6	关于何时用指针传递参数	33
原则 3.7	避免使用参数不确定的函数	34
原则 3.8	若不得不使用参数不确定的函数，用 <stdarg.h> 提供的方法	35
原则 3.9	避免函数的参数过多	36
原则 3.10	尽量保持函数只有唯一出口	36

原则 3.11	显式定义返回类型	36
原则 3.12	(非 void) 任何情况都要有返回值	37
原则 3.13	若函数返回状态, 尝试用枚举作类型	37
原则 3.14	返回指针类型的函数应该用 NULL 表示失败	38
原则 3.15	函数尽量返回引用 (而不是值)	38
原则 3.16	若必须返回值, 不要强行返回引用	39
原则 3.17	当函数返回引用或指针时, 用文字描述其有效期	40
原则 3.18	禁止成员函数返回成员 (可读写) 的引用或指针	40
原则 3.19	重复使用的代码用函数替代	41
原则 3.20	关于虚友元函数	41
原则 3.21	关于虚构造函数	43
第 4 章	类的设计和声明	45
原则 4.1	类应是描述一组对象的集合	45
原则 4.2	类成员应是私有的 (private)	45
原则 4.3	保持对象状态信息的持续性	46
原则 4.4	提高类内聚合度	46
原则 4.5	降低类间的耦合度	47
原则 4.6	努力使类的接口少而完备	48
原则 4.7	保持类的不同接口在实现原则上的一致性	48
原则 4.8	保持不同类的接口在实现原则上的一致性	49
原则 4.9	避免为每个类成员提供访问函数	50
原则 4.10	不要在类定义时提供成员函数体	50
原则 4.11	函数声明 (而不是实现) 时定义参数的缺省值	51
原则 4.12	恰当选择成员函数、全局函数和友元函数	51
原则 4.13	防范、杜绝潜在的二义性	52
原则 4.14	显式禁止编译器自动生成不需要的函数	53
原则 4.15	当遇到错误时对象应该应对有度	54
原则 4.16	用嵌套类的方法减少匿名命名空间类的数量	54
第 5 章	(面向对象的) 继承	55
原则 5.1	“公共继承 (public inheritance)” 意味着 “派生类是基类”	55
原则 5.2	关于 “有” 和 “由...实现”	55

原则 5.3	关于继承和模板 (template) 的区别	56
原则 5.4	关于继承接口和继承实现	57
原则 5.5	限制继承的层数	59
原则 5.6	继承树上非叶子节点的类型应是虚基类	59
原则 5.7	显式提供继承和访问修饰: public、protected 或 private	60
原则 5.8	显式指出继承的虚函数	61
原则 5.9	基类析构函数 (destructor) 首选是虚函数	62
原则 5.10	绝不要重新定义 (继承来的) 非虚函数	62
原则 5.11	绝不要重新定义缺省参数值	63
原则 5.12	不要将基类强制转换成派生类	64
原则 5.13	关于 C++ 中的分支用法选择	64
原则 5.14	慎用多重继承	65
原则 5.15	所有多重继承的基类析构函数都应是虚函数	66
第 6 章 内存分配和释放		67
原则 6.1	用 new、delete 取代 malloc、calloc、realloc 和 free	67
原则 6.2	new、delete 和 new[]、delete[] 要成对使用	67
原则 6.3	确保所有 new 出来的东西适时被 delete 掉	68
原则 6.4	谁申请谁释放	68
原则 6.5	当对象消亡时确保指针成员指向的系统堆内存全部被释放	70
原则 6.6	自定义类的 new/delete 操作符一定要符合原操作符的行为规范	72
原则 6.7	自定义类的 new 操作符一定要自定义类的 delete 操作符	73
原则 6.8	当所指的内存被释放后, 指针应有一个合理的值	73
原则 6.9	记住给字符串结束符申请空间	74
第 7 章 初始化和清除		75
原则 7.1	声明后就初始化强于使用前才初始化	75
原则 7.2	初始化要彻底	76
原则 7.3	确保每一个构造函数都实现完全的初始化	76
原则 7.4	尽量使用初始化列表	77
原则 7.5	初始化列表要按成员声明顺序初始化它们	78
原则 7.6	构造函数没结束, 对象就没有构造出来	79
原则 7.7	不要用构造函数初始化静态成员	79

原则 7.8	拷贝构造函数和赋值函数尽量用常量参数	79
原则 7.9	让赋值函数返回当前对象的引用	80
原则 7.10	在赋值函数中防范自己赋值自己	80
原则 7.11	拷贝和赋值要确保彻底	81
原则 7.12	关于构造函数、析构函数、赋值函数、相等或不等函数的格式	82
原则 7.13	为大多数类提供缺省和拷贝构造函数、析构函数、赋值函数、相等函数	83
原则 7.14	只有在有意义时才提供缺省构造函数	83
原则 7.15	包含资源管理的类应自定义拷贝构造函数、赋值函数和析构函数	84
原则 7.16	拷贝构造函数、赋值函数和析构函数要么全自定义，要么全生成	84
原则 7.17	类应有自己合理的拷贝原则：或浅拷贝或深拷贝	84
原则 7.18	若编译时会完全初始化，不要给出数组的尺寸	85
原则 7.19	将循环索引的初值定在循环点附近	85
原则 7.20	确保全局变量在使用前被初始化	86
第 8 章 常量		89
原则 8.1	关于常量修饰符的含义	89
原则 8.2	在设计函数原型时，对那些不可能被修改的参数用常量修饰	89
原则 8.3	类成员可以转换成常量形式暴露出来	90
原则 8.4	关于常量成员函数	91
原则 8.5	不要让常量成员函数修改程序的状态	92
原则 8.6	不要将常量强制转换成非常量	92
原则 8.7	任何变量和成员函数，首选用 <code>const</code> 修饰	92
第 9 章 重载		93
原则 9.1	仔细区分带缺省值参数的函数和重载函数	93
原则 9.2	确保重载函数的所有版本有共同的目的和相似的行为	94
原则 9.3	避免重载在指针和整型类型上	94
原则 9.4	尽量避免重载在模板类型上	95
第 10 章 操作符		97
原则 10.1	遵守操作符原本的含义，不要创新	98

原则 10.2	确保自定义操作符能和其他操作符混合使用	98
原则 10.3	区分作为成员函数和作为友元的操作符	98
原则 10.4	关于前后缀操作符	99
原则 10.5	确保相关的一组操作符行为统一	100
原则 10.6	绝不要自定义 <code>operator&&()</code> 、 <code>operator ()</code> 和 <code>operator,()</code>	100
第 11 章	类型转换	103
原则 11.1	尽量避免强制类型转换	103
原则 11.2	如果不得不做类型转换, 尽量用显式方式	103
原则 11.3	使用新型的类型转换并确保选择正确	104
原则 11.4	用虚函数方式取代 <code>dynamic_cast</code>	104
原则 11.5	自定义类最好提供显式而不是隐式转换函数	105
原则 11.6	用关键字 <code>explicit</code> 防止单参数构造函数的类型转换功能	106
原则 11.7	限制隐式类型转换的类型数	107
原则 11.8	避免多个函数提供相同的类型转换	107
第 12 章	友元	109
原则 12.1	少用友元	109
原则 12.2	减少拥有友元特权的个数	109
第 13 章	模板	111
原则 13.1	使用模板如果有限制条件一定要在注释和文档中描述清楚	111
原则 13.2	模板类型应传引用/指针而不是值	112
原则 13.3	注意模板编译的特殊性	112
原则 13.4	嵌套 <code>template</code> 的 <code>>></code> 中间要加空格以区别于 <code>operator>></code>	114
第 14 章	表达式和控制流程	115
原则 14.1	让表达式直观	115
原则 14.2	避免在表达式中用赋值语句	115
原则 14.3	不能将枚举类型进行运算后再赋给枚举变量	116
原则 14.4	避免对浮点类型做等于或不等于判断	116