

<<Effective C++中文版>>

图书基本信息

书名：<<Effective C++中文版>>

13位ISBN编号：9787560925257

10位ISBN编号：7560925251

出版时间：2001-9

出版时间：华中科技大学出版社

作者：[美] Scott Meyers

页数：259

译者：侯捷

版权说明：本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问：<http://www.tushu007.com>

<<Effective C++中文版>>

内容概要

您手上这本书，是世界顶级C++大师Scott Meyers成名之作的第二版。

其第一版诞生于1991年。

在国际上，本书所引起的反响之大，波及整个计算机技术出版领域，余音至今未绝。

几乎在所有C++书籍的推荐名单上，本书都会位于前三名。

作者高超的技术把握力，独特的视角、诙谐轻松的写作风格、独具匠心的内容组织，都受到极大的推崇和仿效。

甚至连本书简洁明快的命名风格，也有着一种特殊的号召力，我可以轻易列举出一大堆类似名字，比如Meyers本人的More Effective C++和Effective STL，Don Box的Effective COM，Stan Lippman主编的Efficient C++系列，Herb Sutter的Exceptional C++等等。

要知道，这可不是出版社的有意安排，而且上面这些作者，同样是各自领域里的绝顶大师，决非人云亦云、欺世盗名之辈。

这种奇特的现象，只能解释为人们对这本书衷心的赞美和推崇。

????然而这样一本掷地有声的C++世界名著，不仅迟迟未能出版简体中文版，而且在国内其声誉似乎也并不显赫。

可以说在一年之前，甚至很少有C++的学习者听说过这本书，这实在是一种遗憾。

今天，在很多人的辛勤努力之下，这本书终于能够展现在我们的面前，对于真正的C++程序员来说，这确实是一件值得弹冠相庆的事。

???我是一名普通的C++爱好者，因为机缘巧合，有幸参与了这本书的繁简转译工作，这使我能够比较早地看到本书的原版和繁体中文版。

在这里我必须表达对本书中文译者、台湾著名技术作家侯捷先生的敬意和感谢，因为在我看来，这本书的中文版在质量上较其英文版兄长分毫不差，任何人都知道，达到这一点是多么的困难。

侯先生以其深厚的技术功底、卓越的语言能力和严谨细致的治学态度，为我们跨越了语言隔阂所带来的理解障碍，完整而生动地将原书的内容与精神表达无遗，更令人钦佩的是，中文版的行文风格与原文也达到了高度的统一，可谓神形兼备，实在令人赞叹！

因此我非常乐意向大家推荐这本书，相信它会在带给您带给你技术享受的同时，也带给您阅读的享受。

????曾经在网络讨论组中间看到这样的说法，C++程序员可以分成两类，读过Effective C++的和没读过的。

或许有点夸张了，但无论如何，当您拥有这本书之后，就获得了迅速提升自己C++功力的一个契机。

这本书不是读完一遍就可以束之高阁的快餐读物，也不是能够立刻解决手边问题的参考手册，而是需要您去反复阅读体会，极力融入自己思想之中，融入自己每一次敲击键盘的动作之中。

C++是真正程序员的语言，背后有着精深的思想与无以伦比的表达能力，这使得它具有类似宗教般的魅力。

希望这本书能够帮助您跨越C++的重重险阻，领略高处才有的壮美，做一个成功而快乐的C++程序员。

书籍目录

译序 (侯捷) v目录 (Contents) vii前言 (Preface) xiii致谢 (Acknowledgments. 中文版略) xvii导读 (Introduction) 改变旧有的C习惯 (Shifting from C to C++) 条款1: 尽量以 const 和 inline 取代 #define Prefer const and inline to #define. 条款2: 尽量以 取代 Prefer to. 条款3: 尽量以 new 和 delete 取代 malloc () 和 free () Prefer new and delete to malloc and free. 条款4: 尽量使用 C++ 风格的注释形式 Prefer C++-style comments. 内存管理 (Memory Management) 条款5: 使用相同形式的新和 delete Use the same form in corresponding uses of new and delete. 条款6: 记得在 destructor 中以 delete 对付 pointer members Use delete on pointer members in destructors. 条款7: 为内存不足的状况预做准备 Be prepared for out-of-memory conditions. 条款8: 撰写 operator new 和 operator delete 时应遵行公约 Adhere to convention when writing operator new and operator delete. 条款9: 避免遮掩了 new 的正规形式 Avoid hiding the "normal" form of new. 条款10: 如果你写了一个 operator new, 请对应也写一个 operator delete Write operator delete if you write operator new. 构造函数、析构函数和 Assignment 运算符 Constructors, Destructors, and Assignment Operators 条款11: 如果 classes 内动态配置有内存, 请为此 class 宣告一个 copy constructor 和一个 assignment 运算符 Declare a copy constructor and an assignment operator for classes with dynamically allocated memory. 条款12: 在 constructor 中尽量以 initialization 取代 assignment Prefer initialization to assignment in constructors. 条款13: Initialization list 中的 members 初始化排列次序应该和其在 class 内的声明次序相同 List members in an initialization list in the order in which they are declared. 条款14: 总是让 base class 拥有 virtual destructor Make sure base classes have virtual destructors. 条款15: 令 operator= 传回 " *this 的 reference " Have operator= return a reference to *this. 条款16: 在 operator= 中为所有的 data members 赋予内容 Assign to all data members in operator=. 条款17: 在 operator= 中检查是否 " 自己赋值给自己 " Check for assignment to self in operator=. 类与函数之设计和声明 Classes and Functions: Design and Declaration 条款18: 努力让接口完满且最小化 Strive for class interfaces that are complete and minimal. 条款19: 区分 member functions, non-member functions 和 friend functions 三者 Differentiate among member functions, non-member functions, and friend functions. 条款20: 避免将 data members 放在公开接口中 Avoid data members in the public interface. 条款21: 尽可能使用 const Use const whenever possible. 条款22: 尽量使用 pass-by-reference (传址), 少用 pass-by-value (传值) Prefer pass-by-reference to pass-by-value. 条款23: 当你必须传回 object 时, 不要尝试传回 reference Don't try to return a reference when you must return an object. 条款24: 在函数重载 (function overloading) 和参数缺省化 (parameter defaulting) 之间, 谨慎抉择 Choose carefully between function overloading and parameter defaulting. 条款25: 避免对指针型别和数值型别进行重载 Avoid overloading on a pointer and a numerical type. 条款26: 防卫潜伏的 ambiguity (模棱两可) 状态 Guard against potential ambiguity. 条款27: 如果不想使用编译器暗自产生的 member functions, 就应该明白拒绝它 Explicitly disallow use of implicitly generated member functions you don't want. 条款28: 尝试切割 global namespace (全域命名空间) Partition the global namespace. 类与函数之实现 Classes and Functions: Implementation 条款29: 避免传回内部数据的 handles Avoid returning " handles " to internal data. 条款30: 避免写出 member function, 传回一个 non-const pointer 或 reference 并以之指向较低存取层级的 members Avoid member functions that return non-const pointers or references to members less accessible than themselves. 条款31: 千万不要传回 " 函数内 local 对象的 reference ", 或是 " 函数内以 new 获得的指针所指向的对象 " Never return a reference to a local object or to a dereferenced pointer initialized by new within the function. 条款32: 尽可能延缓变量定义式的出现 Postpone variable definitions as long as possible. 条款33: 明智地运用 inlining Use inlining judiciously. 条款34: 将文件之间的编译依赖关系 (compilation dependencies) 降至最低 Minimize compilation dependencies between files. 继承机制与面向对象设计 Inheritance and Object-Oriented Design 条款35: 确定你的 public inheritance 模塑出 " isa " 的关系 Make sure public inheritance models " isa. " 条款36: 区分 " 接口继承 (interface inheritance) " 和 " 实现继承 (implementation inheritance) " Differentiate between inheritance of interface and inheritance of implementation. 条款37: 绝对不要重新定义一个继承而来的非虚拟函数 Never redefine an inherited nonvirtual function. 条款38: 绝对不要重新定义一个继承而来的缺省参数值 Never redefine an inherited

<<Effective C++中文版>>

default parameter value. 条款39：避免在继承体系中做 cast down（向下转型）的动作 Avoid casts down the inheritance hierarchy. 条款40：通过 layering（分层技术）来模塑 has-a 或 is-implemented-in-terms-of 的关系 Model “has-a” or “is-implemented-in-terms-of” through layering. 条款41：区分 inheritance 和 templates Differentiate between inheritance and templates. 条款42：明智地运用 private inheritance（私有继承） Use private inheritance judiciously. 条款43：明智地运用多继承（multiple inheritance, MI） Use multiple inheritance judiciously. 条款44：说出你的意思并了解你所说的每一句话 Say what you mean； understand what you're saying. 杂项讨论（Miscellany） 条款45：知道 C++（编译器）默默为我们完成和调用哪些函数 Know what functions C++ silently writes and calls. 条款46：宁愿编译和连接时出错，也不要运行时才错 Prefer compile-time and link-time errors to runtime errors. Ensure that non-local static objects are initialized before they're used. 条款48：不要对编译器的警告讯息视而不见 Pay attention to compiler warnings. 条款49：尽量让自己熟悉 C++ 标准程序库 Familiarize yourself with the standard library. 条款50：加强自己对 C++ 的了解 Improve your understanding of C++

<<Effective C++中文版>>

媒体关注与评论

本书的50条准则，每一条都扼要说明了一个可让你写出更好的C++ 程序代码的方法，并以特别设计过的例子详加讨论。

在些新版中，Meyers重新检验了每一准则，并特别注意兼容于C++ 标准规格与现行编译器技术，及融入软件界对C++ 运用之最新观察结果。

编辑推荐

我是一名普通的C++爱好者，因为机缘巧合，有幸参与了这《Effective C++中文版》的繁简转译工作，这使我能够比较早地看到《Effective C++中文版》的原版和繁体中文版。

在这里我必须表达对《Effective C++中文版》中文译者、台湾著名技术作家侯捷先生的敬意和感谢，因为在我看来，这《Effective C++中文版》的中文版在质量上较其英文版兄长分毫不差，任何人都知道，达到这一点是多么的困难。

侯先生以其深厚的技术功底、卓越的语言能力和严谨细致的治学态度，为我们跨越了语言隔阂所带来的理解障碍，完整而生动地将原书的内容与精神表达无遗，更令人钦佩的是，中文版的行文风格与原文也达到了高度的统一，可谓神形兼备，实在令人赞叹！

因此我非常乐意向大家推荐这《Effective C++中文版》，相信它会在带给您带给你技术享受的同时，也带给您阅读享受。

曾经在网络讨论组中间看到这样的说法，C++程序员可以分成两类，读过Effective C++的和没读过的。

或许有点夸张了，但无论如何，当您拥有这《Effective C++中文版》之后，就获得了迅速提升自己C++功力的一个契机。

这《Effective C++中文版》不是读完一遍就可以束之高阁的快餐读物，也不是能够立刻解决手边问题的参考手册，而是需要您去反复阅读体会，极力融入自己思想之中，融入自己每一次敲击键盘的动作之中。

C++是真正程序员的语言，背后有着精深的思想与无以伦比的表达能力，这使得它具有类似宗教般的魅力。

希望这《Effective C++中文版》能够帮助您跨越C++的重重险阻，领略高处才有的壮美，做一个成功而快乐的C++程序员。

<<Effective C++中文版>>

版权说明

本站所提供下载的PDF图书仅提供预览和简介，请支持正版图书。

更多资源请访问:<http://www.tushu007.com>