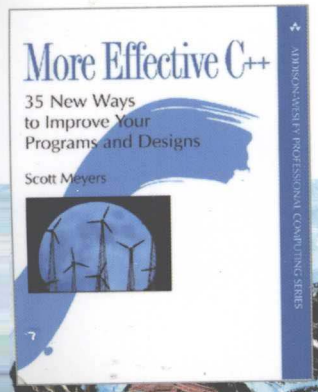
Broadview
www.broadview.com.cn
PEARSON

传世经典书丛
Eternal Classics

More Effective C++

35个改善编程与设计的有效方法 中文版



[美] Scott Meyers 著
侯捷 译

電子工業出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
http://www.phei.com.cn

More Effective C++ 中文版

35个改善编程与设计的有效方法

More Effective C++:
35 New Ways to Improve Your Programs and Designs

[美] Scott Meyers 著
侯 捷 译

電子工業出版社
Publishing House of Electronics Industry
北京•BEIJING

内 容 简 介

继 *Effective C++*之后，Scott Meyers 于 1996 推出这本“续集”。条款变得比较少，页数倒是多了一些，原因是这次选材比“第一集”更高阶，尤其是第 5 章。Meyers 将此章命名为技术（techniques），并明白告诉你，其中都是一些 patterns，例如 virtual constructors，smart pointers，reference counting，proxy classes，double dispatching……这一章的每个条款篇幅都达 15~30 页之多，实在让人有“山重水复疑无路，柳暗花明又一村”之叹。

虽然出版年代稍嫌久远，但本书并没有第 2 版，原因是当其出版之时（1996），C++ Standard 已经几乎定案，本书即依当时的标准草案而写，其与现今的C++标准规范几乎相同。而且可能变化的几个弹性之处，Meyers也都有所说明与提示。读者可以登录作者提供的网址，看看上下两集的勘误与讨论（数量之多，令人惊恐。幸好多是技术讨论或文字斟酌，并没有什么重大误失）。

版权贸易合同登记号　图字：01-2010-7892

图书在版编目（CIP）数据

More Effective C++：35 个改善编程与设计的有效方法/（美）梅耶（Meyers,S.）著；侯捷译. 北京：电子工业出版社，2011.1
（传世经典书丛）
书名原文：More Effective C++:35 New Ways to Improve Your Programs and Designs
ISBN 978-7-121-12570-6

Ⅰ. ①M… Ⅱ. ①梅… ②侯… Ⅲ. ①C 语言－程序设计 Ⅳ. ①TP312

中国版本图书馆 CIP 数据核字（2010）第 247371 号

策划编辑：张春雨
责任编辑：付　睿　李云静
印　　刷：
装　　订：北京市铁成印刷厂
出版发行：电子工业出版社
　　北京市海淀区万寿路 173 信箱　　邮编：100036
开　　本：787×980　1/16　印张：21　字数：500 千字
印　　次：2011 年 1 月第 1 次印刷
定　　价：59.00 元

凡所购买电子工业出版社图书有缺损问题，请向购买书店调换。若书店售缺，请与本社发行部联系，联系及邮购电话：（010）88254888。

质量投诉请发邮件至 zlts@phei.com.cn，盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线：（010）88258888。

悦读上品 得乎益友

孔子云："取乎其上，得乎其中；取乎其中，得乎其下；取乎其下，则无所得矣"。

对于读书求知而言，这句古训教我们去读好书，最好是好书中的上品——经典书。其中，科技人员要读的技术书，因为直接关乎客观是非与生产效率，阅读选材本更应慎重。然而，随着技术图书品种的日益丰富，发现经典书越来越难，尤其对于涉世尚浅的新读者，更为不易，而他们又往往是最需要阅读、提升的重要群体。

所谓经典书，或说上品，是指选材精良、内容精练、讲述生动、外延丰盈、表现手法体贴入微的读品，它们会成为读者的知识和经验库中的重要组成部分，并且拥有从不断重读中汲取养分的空间。因此，选择阅读上品的问题便成了有效阅读的首要问题。当然，这不只是效率问题，上品促成的既是对某一种技术、思想的真正理解和掌握，同时又是一种感悟或享受，是一种愉悦。

与技术本身类似，经典 IT 技术书多来自国外。深厚的积累、良好的写作氛围，使一批大师为全球技术学习者留下了璀璨的智慧瑰宝。就在那个年代即将远去之时，无须回眸，也能感受到这一部部厚重而深邃的经典著作，在造福无数读者后从未蒙尘的熠熠光辉。而这些凝结众多当今国内技术中坚美妙记忆与绝佳体验的技术图书，虽然尚在国外图书市场上大放异彩，却已逐渐淡出国人的视线。最为遗憾的是，迟迟未有可以填补空缺的新书问世。而无可替代，不正是经典书被奉为圭臬的原因？

为了不让国内读者，尤其是即将步入技术生涯的新一代读者，就此错失这些滋养过先行者们的好书，以出版 IT 精品图书，满足技术人群需求为己任的我们，愿意承担这一使命。本次机遇惠顾了我们，让我们有机会携手权威的 Pearson 公司，精心推出"传世经典书丛"。

在我们眼中，"传世经典"的价值首先在于——既适合喜爱科技图书的读者，也符合专家们挑剔的标准。幸运的是，我们的确找到了这些堪称上品的佳作。丛书带给我们的幸运颇多，细数一下吧。

得以引荐大师著作

有恐思虑不周，我们大量参考了国外权威机构和网站的评选结果，并得到了 Pearson 的专业支持，又进

一步对符合标准之图书的国内外口碑与销售情况进行细致分析，也听取了国内技术专家的宝贵建议，才有幸选出对国内读者最富有技术养分的大师上品。

向深邃的技术内涵致敬

中外技术环境存在差异，很多享誉国外的好书未必适用于国内读者；且技术与应用瞬息万变，很容易让人心生迷惘或疲于奔命。本丛书的图书遴选，注重打好思考方法与技术理念的根基，旨在帮助读者修炼内功，提升境界，将技术真正融入个人知识体系，从而可以一通百通，从容面对随时涌现的技术变化。

翻译与评注的双项选择

引进优秀外版著作，将其翻译为中文供国内读者阅读，较为有效与常见。但另有一些外语水平较高、喜好阅读原版的读者，苦于对技术理解不足，不能充分体会原文表述的精妙，需要有人指导与点拨。而一批本土技术精英经过长期经典熏陶及实践锤炼，已足以胜任这一工作。有鉴于此，本丛书在翻译版的同时推出融合英文原著与中文点评、注释的评注版，供不同志趣的读者自由选择。

承蒙国内一流译(注)者的扶持

优秀的英文原著最终转化为真正的上品，尚需跨越翻译鸿沟，外版图书的翻译质量一直屡遭国内读者诟病。评注版的增值与含金量，同样依赖于评注者的高卓才具。好在，本丛书得到了久经考验的权威译(注)者的认可和支持，首肯我们选用其佳作，或亲自参与评注工作。正是他们的参与保证了经典的品质，既再次为我们的选材把关，更提供了一流的中文表述。

期望带给读者良好的阅读体验

一本好书带给人的愉悦不止于知识收获，良好的阅读感受同样不可缺少，且对学业不无助益。为让读者收获与上品相称的体验，我们在图书装帧设计与选材用料上同样不敢轻率，惟愿送到读者手中的除了珠玑章句，还有舒适与熨帖的视觉感受。

所有参与丛书出版的人员，尽管能力有限，却无不心怀严谨之心与完美愿望。如果读者朋友能从潜心阅读这些上品中偶有获益，不啻为对我们工作的最佳褒奖。若有阅读感悟，敬请拨冗告知，以鼓励我们继续在这一道路上贡献绵薄之力。如有不周之处，也请不吝指教。

电子工业出版社博文视点

二〇一〇年十二月

More Effective C++ 的荣耀：

这是一本多方面发人深省的 C++ 书籍：不论在你偶尔用到的语言特性上，或是在你自以为十分熟悉的语言特性上。只有深刻了解 C++ 编译器如何解释你的代码，你才有可能用 C++ 语言写出健壮的软件。本书是协助你获得此等层级的了解过程中，一份极具价值的资源。读过本书之后，我感觉像是浏览了 C++ 编程大师所检阅过的代码，并获得许多极具价值的洞见。

——Fred Wild, Vice President of Technology,
Advantage Software Technologies

本书内含大量重要的技术，这些技术是撰写优良 C++ 程序所不可或缺的。本书解释如何设计和实现这些观念，以及潜伏在其他某些替代方案中的陷阱。本书亦含晚些加入的 C++特性的详细说明。任何人如果想要好好地运用这些新特性，最好买一本并且放在随手可得之处，以备查阅。

——Christopher J. Van Wyk, Professor
Mathematics and Computer Science, Drew University

这是一本具备工业强度的最佳书籍。对于已经阅读过 *Effective C++* 的人，这是完美的续集。

——Eric Nagler, C++ Instructor and Author,
University of California Santa Cruz Extension

More Effective C++ 是一本无微不至且很有价值的书籍，是 Scott 第一本书 *Effective C++* 的续集。我相信每一位专业的 C++ 软件开发人员都应该读过并记忆 *Effective C++* 和 *More Effective C++* 两本书内的各种招式，以及其中重要（而且有时候不可思议）的语言的方方面面。我强烈推荐这两本书给软件开发人员、测试人员、管理人员……，每个人都可以从 Scott 专家级的知识与卓越的表达能力中获益。

——Steve Burkett, Software Consultant

For Clancy,
my favorite enemy within.

献给每一位对 C++/OOP 有所渴望的人

正确的观念 重于一切

– 侯 捷 –

译序

C++ 是一门难学易用的语言！

C++ 的难学，不仅在其广博的语法、语法背后的语义、语义背后的深层思维、深层思维背后的对象模型；C++ 的难学，还在于它提供了 4 种不同（相辅相成）的编程思维模型：procedural-based，object-based，object-oriented，generic paradigm。

世上没有白吃的午餐。又要有效率，又要有弹性，又要前瞻望远，又要回溯相容，又要能治大国，又要能烹小鲜，学习起来当然就不可能太简单。

在如此庞大复杂的机制下，万千使用者前赴后继的动力是：一旦学成，妙用无穷。

C++ 相关书籍之多，车载斗量，如天上繁星，如过江之鲫。广博如四库全书者有之（*The C++ Programming Language*、*C++ Primer*），深奥如重山复水者有之（The *Annotated C++ Reference Manual*, *Inside the C++ Object Model*），细说历史者有之（*The Design and Evolution of C++*, *Ruminations on C++*），独沽一味者有之（*Polymorphism in C++*, *Genericity in C++*），独树一帜者有之（*Design Patterns*, *Large Scale C++ Software Design*, *C++ FAQs*），程序库大全有之（*The C++ Standard Library*），另辟蹊径者有之（*Generic Programming and the STL*），工程经验之累积亦有之（*Effective C++*, *More Effective C++*, *Exceptional C++*）。

这其中，“工程经验之累积”对已具 C++ 相当基础的程序员而言，有着致命的吸引力与立竿见影的帮助。Scott Meyers 的 *Effective C++* 和 *More Effective C++* 是此类佼佼，Herb Sutter 的 *Exceptional C++* 则是后起之秀。

这类书籍的一个共同特色是轻薄短小，并且高密度地纳入作者浸淫于 C++/OOP 领域多年而广泛的经验。它们不但开扩读者的视野，也为读者提供各种

C++/OOP 常见问题或易犯错误的解决模型。某些小范围主题诸如“在 base classes 中使用 virtual destructor”、“令 operator= 传回 *this 的 reference”，可能在百科型 C++ 语言书籍中亦曾概略提过，但此类书籍以深度探索的方式，让我们了解问题背后的成因、最佳的解法，以及其他可能的牵扯。至于大范围主题，例如 smart pointers，reference counting，proxy classes，double dispatching，基本上已属 design patterns 的层级！

这些都是经验的累积和心血的结晶！

我很高兴将以下两本优秀书籍，规划为一个系列，以郑重的形式呈现给您：

1. *Effective* C++ 2/e, by Scott Meyers, AW 1998
2. *More Effective* C++, by Scott Meyers, AW 1996

本书不但与英文版页页对译，保留索引，并加上译注、交叉索引[1]、读者服务[2]。

这套书将对于您的程序设计生涯带来重大帮助。翻译这套书籍的过程中，我感觉来自技术体会上的极大快乐。我祈盼（并相信）您在阅读此书时拥有同样的心情。

侯捷 2003/03/07 于台湾新竹
jjhou@jjhou.com
http://www.jjhou.com

[1] *Effective* C++ 2/e 和 *More Effective* C++ 之中译本，实际上是以 Scott Meyers 的另一个产品 *Effective* C++ *CD* 为本，不仅数据更新，同时亦将 CD 版中的两书交叉参考保留下来，可为读者带来旁征博引时的莫大帮助。

[2] 欢迎读者对本书所及主题提出讨论，并感谢读者对本书任何失误提出指正。来信请寄 jjhou@jjhou.com。勘误网站：http://www.jjhou.com（繁），http://jjhou.csdn.net（简）

本书保留大量简短易读之英文术语，时而中英并陈。以下用语请读者特别注意：

英文术语	本书译词	英文术语	本书译词
argument	自变量 （i.e. 实参）	instantiated	实例化、具现化
by reference	传址	library	程序库
by value	传值	resolve	决议
dereference	解引（i.e. 解参考）	parameter	参数 （i.e. 形参）
evaluate	评估、核定	type	型别 （i.e. 类型）
instance	实例		

译注：借此版面提醒读者，本书之中如果出现“条款 5”这样的参考指示，指的是本书条款 5；如果出现“条款 **E**5”这样的参考指示，**E** 是指 *Effective C++ 2/e*）

目录

导读
Introduction

对 C++ 程序员而言，日子似乎有点过于急促。虽然只商业化不到 10 年，C++ 却俨然成为几乎所有主要计算环境的系统程序语言霸主。面临程序设计方面极具挑战性问题的公司和个人，不断投入 C++ 的怀抱。而那些尚未使用 C++ 的人，最常被询问的一个问题则是：你打算什么时候开始用 C++。C++ 标准化已经完成，其所附带的标准程序库幅员广大，不仅涵盖 C 函数库，也使之相形见绌。这么一个大型程序库使我们有可能在不必牺牲移植性的情况下，或是在不必从头撰写常用算法和数据结构的情况下，完成琳琅满目的各种复杂程序。C++ 编译器的数量不断增加，它们所供应的语言性质不断扩充，它们所产生的代码质量也不断改善。C++ 开发工具和开发环境愈来愈丰富，威力愈来愈强大，健壮（robust）的程度也愈来愈高。商业化程序库几乎能够满足各个应用领域中的编程需求。

一旦语言进入成熟期，我们对它的使用经验也就愈来愈多，我们所需要的信息也就随之改变了。1990 年人们想知道 C++ 是什么东西。到了 1992 年，他们想知道如何运用它。如今 C++ 程序员问的问题更高级：我如何能够设计出适应未来需求的软件？我如何能够改善代码的效率而不折损正确性和易用性？我如何能够实现语言不能直接支持的精巧功能？

这本书中我要回答这些问题，以及其他许多类似问题。

本书将告诉你如何更具实效地设计并实现 C++ 软件：让它行为更正确，面对异常时更健壮、更有效率、更具移植性，将语言特性发挥得更好，更优雅地调整适应，在“混合语言”开发环境中运作得更好，更容易被正确运用，更不容易被误用。简单地说就是如何让软件更好。

本书内容分为 35 个条款。每个条款都在特定主题上精简摘要出 C++ 程序设计社区所积累的智能。大部分条款以准则的形式呈现，随附的说明则阐述这条准则为什么存在，如果不遵循会发生什么后果，以及什么情况下可以合理违反该准则。

所有条款被我分为数大类。某些条款关心特定的语言性质，特别是你可能少有使用经验的一些新性质。例如条款 9~15 专注于 exceptions（就像 Tom Cargill，Jack Reeves，Herb Sutter 所发表的那些杂志文章一样）。其他条款解释如何结合语言的不同特性以达成更高阶目标。例如条款 25~31 描述如何限制对象的个数或诞生地点，如何根据一个以上的对象类型产生出类似虚函数的东西，如何产生 smart pointers 等。其他条款解决更广泛的题目。条款 16~24 专注于效率上的议题。不论哪一个条款，提供的都是与其主题相关且意义重大的做法。在 *More Effective C++* 一书中你将学习到如何更实效、更精锐地使用 C++。大部分 C++ 教科书中对语言性质的大量描述，只能算是本书的一个背景信息而已。

这种处理方式意味着，你应该在阅读本书之前便熟悉 C++。我假设你已了解 classes（类）、保护层级（protection levels）、虚函数、非虚函数，我也假设你已通晓 templates 和 exceptions 背后的概念。我并不期望你是一位语言专家，所以涉及较罕见的 C++ 特性时，我会进一步解释。

本书所谈的 C++

我在本书所谈、所用的 C++，是 ISO/ANSI 标准委员会于 1997 年 11 月完成的 C++ 国际标准最后草案（Final Draft International Standard）。这暗示了我所使用的某些语言特性可能并不在你的编译器（s）支持能力之列。别担心，我认为对你而言唯一所谓的“新”特性，应该只有 templates，而 templates 如今几乎已是各家编译器的必备功能。我也运用 exceptions，并大量集中于条款 9~15。如果你的编译器（s）未能支持 exceptions，没什么大不了，这并不影响本书其他部分带给你的好处。但是，听我说，即使你不需要用到 exceptions，亦应阅读条款 9~15，因为那些条款（及其相关篇幅）检验了某些不论什么场合下你都应该了解的主题。

我承认，就算标准委员会授意某一语言特性或是赞同某一实务做法，并非就保证该语言特性已出现在目前的编译器上，或该实务做法已可应用于既有的开发环境上。

一旦面对“标准委员会所议之理论”和“真正能够有效运作之实务”间的矛盾，我便将两者都加以讨论，虽然我其实更重视实务。由于两者我都讨论，所以当你的编译器（s）和 C++ 标准不一致时，本书可以协助你，告诉你如何使用目前既有的架构来模拟编译器（s）尚未支持的语言特性。而当你决定将一些原本绕道而行的解决办法以新支持的语言特性取代时，本书亦可引导你。

注意当我说到编译器（s）时，我使用复数。不同的编译器对 C++ 标准的满足程度各不相同，所以我鼓励你在至少两种编译器（s）平台上发展代码。这么做可以帮助你避免不经意地依赖某个编译器专属的语言延伸性质，或是误用某个编译器对标准规格的错误阐释。这也可以帮助你避免使用过度先进的编译器技术，例如，独家厂商才做得出的某种语言新特性。如此特性往往实现得不够精良（臭虫多，要不就表现得迟缓，或是两者兼具），而且 C++ 社区往往对这些特性缺乏使用经验，无法给你应用上的忠告。雷霆万钧之势固然令人兴奋，但当你的目标是要产生可靠的代码，恐怕还是步步为营（并且能够与人合作）的好。

本书用了两个你可能不太熟悉的 C++ 性质，它们都是晚些才加入 C++ 标准之中的。某些编译器支持它们，但如果你的编译器不支持，你可以轻易地用你所熟悉的其他性质来模拟它们。

第一个性质是 bool 类型，其值必为关键词 true 或 false。如果你的编译器尚未支持 bool，有两个方法可以模拟它。第一个方法是使用一个 global enum：

```
enum bool { false, true };
```

这允许你将参数为 bool 或 int 的不同函数加以重载（overloading）。缺点是，内建的“比较操作符（comparison operators）”如 ==，<，>=，等仍旧返回 ints。所以以下代码的行为不如我们所预期：

```
void f(int);
void f(bool);
int x, y;
...
f( x < y );       // 调用 f(int)，但其实它应该调用 f(bool)。
```

一旦你改用真正支持 bool 的编译器，这种 enum 近似法可能会造成程序行为的改变。

另一种做法是利用 typedef 来定义 bool，并以常量对象作为 true 和 false：

```
typedef int bool;
const bool false = 0;
const bool true = 1;
```

这种手法与传统的 C/C++ 语义兼容。使用这种仿真法的程序，在移植到一个支持有 bool 类型的编译器平台之后，行为并不会改变。缺点则是无法在函数重载（overloading）时区分 bool 和 int。以上两种近似法都有道理，请选择最适合你的一种。

第二个新性质，其实是 4 个转型操作符：static_cast，const_cast，dynamic_cast 和 reinterpret_cast。如果你不熟悉这些转型操作符，请翻到条款 2 仔细阅读其中内容。它们不只比它们所取代的 C 旧式转型做得更多，也更好。书中任何时候当我需要执行转型动作，我都使用新式的转型操作符。

C++ 拥有比语言本身更丰富的东西。是的，C++ 还有一个伟大的标准程序库（见条款 E49）。我尽可能使用标准程序库所提供的 string 类型来取代 char* 指针，而且我也鼓励你这么做。string objects 并不比 char*-based 字符串难操作，它们的好处是可以免除你大部分的内存管理工作。而且如果发生 exception 的话（见条款 9 和 10），string objects 比较不会出现 memory leaks（内存泄漏）问题。实现良好的 string 类型甚至可和对应的 char* 比赛效率，而且可能会赢（条款 29 会告诉你其中的故事）。如果你不打算使用标准的 string 类型，你当然会使用类似 string 的其他 classes，是吧？是的，用它，因为任何东西都比直接使用 char* 来得好。

我将尽可能使用标准程序库提供的数据结构。这些数据结构来自 Standard Template Library（“STL”——见条款 35）。STL 包含 bitsets，vectors，lists，queues，stacks，maps，sets，以及更多东西，你应该尽量使用这些标准化的数据结构，不要情不自禁地想写一个自己的版本。你的编译器或许没有附带 STL 给你，但不要因为这样就不使用它。感谢 Silicon Graphics 公司的热心，你可以从 SGI STL 网站下载一份免费产品，它可以和多种编译器配合使用。