HZ BOOKS

PEARSON Addison Wesley

Modern C++ Design C++ 设计新思维

More Exceptional C++

40 New Engineering Puzzles, Programming Problems, and Solutions

（英文版）

（美） Herb Sutter 著

机械工业出版社
China Machine Press

More Exceptional C++

(英文版)

40 New Engineering Puzzles, Programming Problems, and Solutions

(美) Herb Sutter 著

机械工业出版社
China Machine Press

English reprint edition copyright © 2006 by Pearson Education Asia Limited and China Machine Press.

Original English language title: *More Exceptional C++: 40 New Engineering Puzzles, Programming Problems, and Solutions* (ISBN 0-201-70434-X) by Herb Sutter, Copyright © 2002.

All rights reserved.

Published by arrangement with the original publisher, Pearson Education, Inc., publishing as Addison-Wesley.

For sale and distribution in the People's Republic of China exclusively (except Taiwan, Hong Kong SAR and Macau SAR).

本书英文影印版由Pearson Education Asia Ltd.授权机械工业出版社独家出版。未经出版者书面许可，不得以任何方式复制或抄袭本书内容。

仅限于中华人民共和国境内（不包括中国香港、澳门特别行政区和中国台湾地区）销售发行。

本书封面贴有Pearson Education（培生教育出版集团）激光防伪标签，无标签者不得销售。

版权所有，侵权必究。

本书法律顾问 北京市展达律师事务所

本书版权登记号：图字：01-2006-0526

图书在版编目（CIP）数据

More Exceptional C++（英文版）/（美）萨特（Sutter, S.）著. －北京：机械工业出版社，2006.3

（C++设计新思维）

书名原文：More Exceptional C++: 40 New Engineering Puzzles, Programming Problems, and Solutions

ISBN 7-111-18370-3

Ⅰ. M… Ⅱ. 萨… Ⅲ. 语言－程序设计－英文 Ⅳ. TP312

中国版本图书馆CIP数据核字（2006）第004683号

机械工业出版社（北京市西城区百万庄大街22号 邮政编码 100037）

责任编辑：迟振春

北京牛山世兴印刷厂印刷 · 新华书店北京发行所发行

2006年3月第1版第1次印刷

718mm × 1020mm 1/16 · 18.75印张

印数：0 001-3 000册

定价：36.00元

凡购本书，如有倒页、脱页、缺页，由本社发行部调换

本社购书热线：（010）68326294

“C++设计新思维”丛书前言

自C++诞生尤其是ISO/ANSI C++标准问世以来，以Bjarne Stroustrup为首的C++社群领袖一直不遗余力地倡导采用“新风格”教学和使用C++。事实证明，除了兼容于C的低阶特性外，C++提供的高级特性以及在此基础上发展的各种惯用法可以让我们编写出更加简洁、优雅、高效、健壮的程序。

这些高级特性和惯用法包括精致且高效的标准库和各种“准标准库”，与效率、健壮性、异常安全等主题有关的各种惯用法，以及在C++的未来占据更重要地位的模板和泛型程序设计技术等。它们发展于力量强大的C++社群，并被这个社群中最负声望的专家提炼、升华成一本本精彩的著作。毫无疑问，这些学术成果必将促进C++社群创造出更多的实践成果。

我个人认为，包括操作系统、设备驱动、编译器、系统工具、图像处理、数据库系统以及通用办公软件等在内的基础软件更能够代表一个国家的软件产业发展质量，迄今为止，此类基础性的软件恰好是C++所擅长开发的，因此，可以感性地说，C++的应用水平在一定程度上可以折射出一个国家的软件产业发展水平和健康程度。

前些年国内曾引进出版了一大批优秀的C++书籍，它们拓宽了中国C++程序员的视野，并在很大程度上纠正了长期以来存在于C++的教育、学习和使用方面的种种误解，对C++相关的产业发展起到了一定的促进作用。然而在过去的两年中，随着.NET、Java技术吸引越来越多的注意力，中国软件产业业务化、项目化的状况愈发加剧，擅长于“系统编程”的C++语言的应用领域似有进一步缩减的趋势，这也导致人们对C++的出版教育工作失去了应有的重视。

机械工业出版社华章分社决定继续为中国C++“现代化”教育推波助澜，从2006年起将陆续推出一套“C++设计新思维”丛书。这套丛书秉持精品、高端的理念，其作译者为包括Herb Sutter在内的国内外知名C++技术专家和研究者、教育者，议题紧密围绕现代C++特性，以实用性为主，兼顾实验性和探索性，形式上则是原版影印、中文译著和原创兼收并蓄。每一本书相对独立且交叉引用，篇幅短小却内容深入。作为这套丛书的特邀技术编辑，我衷心希望它们所展示的技术、技巧和理念能够为中国C++社群注入新的活力。

荣　耀
2005年12月
南京师范大学
www.royaloo.com

For Günter and Elisabeth

此为试读,需要完整PDF请访问: www.ertongbook.com

序　言

如何才能成为专家？在我了解的所有领域中答案都是一样的：

1. 学习基础知识。

2. 再一次学习同样的素材，但这一次集中于你初次学习时未意识到其重要性的细节。

如果你选择了适当的细节且对它们的掌握透彻到无需为之费神的地步，你就快要成为一位专家了。然而，在真正成为专家之前，你又如何知道应该选择哪些细节呢？如果有人已经为你挑选好适当的细节，你学习起来将会更快速、更快乐。

例如，我曾经参加过一位不错的摄影师Fred Picker举办的摄影研讨班。他告诉我们，摄影只有两个困难的环节，一是如何取景，二是何时按快门。接下来他花费大部分研讨班时间向我们介绍关于曝光、加工和冲印的技术细节——只有彻底理解了这些细节，我们才能很好地掌握摄影，否则关注那两个“困难”的环节就了无意义。

尝试回答关于C++程序的问题，是一种尤其富有乐趣的学习C++编程细节的方式。例如：

- f(a++);和f(a); ++a;具有相同的效果吗？
- 你能使用一个迭代器来改变一个set的内容吗？
- 假定你正在使用一个名为v的vector，它占用的内存已经增长至让人感到不爽的地步，你希望清除该vector并将其占用的内存归还给系统，那么，v.clear()能胜任吗？

你也许已经猜到这些貌似明显的问题答案必定是“no”，否则我就不会问这些问题了，但你知道答案为何是“no”吗？你确信？

本书回答了这些问题以及其他很多精心挑选的“貌似很普通的程序”的问题。没有多少书籍像本书一样（当然了，它的姊妹篇《Exceptional C++》除外）。很多宣称是“高级”的C++书籍，要么是关于专门的主题（只有当你希望精通这些特定的主题而非试图深入研究日常编程问题时，这些书才有意义），要么纯粹是拿“高级”一词来吸引读者的眼球。

一旦透彻理解了这些问题及其答案，你在编程时就不必过多地烦神于有关细节，大可将精力集中于真正努力解决的那些问题之上。

Andrew Koenig

2001年6月

前　言

古希腊哲学家苏格拉底通过向学生提问进行教学。那些问题被设计用于引导他们，帮助他们从已知的东西得出结论，还向他们展示正在学习的东西是如何相互联系的，又如何与他们已有的知识相互联系。这种教学方式是如此著名，以至于今天我们称之为“苏格拉底教学法”。从我们作为学生的观点来看，苏格拉底的教学方法能够引起我们的兴趣，促使我们思考，帮助我们联系并应用已知的知识去获取新知识。

本书与其姊妹篇《Exceptional C++》[Sutter00]一样，借鉴了苏格拉底的方法。本书假定你正身处编写产品级C++软件的某些领域，它使用问答的形式教你如何有效地使用标准C++及其标准库，并特别关注于在现代C++中实施健全的软件工程。很多问题都直接来源于我和其他人在编写产品级C++代码时所积累的经验。问题的目标在于帮助你从已知的以及刚学到的东西中得出结论，并展示它们之间是如何相互联系的。而谜题则向你展示如何对C++设计和编程问题进行理性的分析。其中有些是常见的议题，有些则不那么常见；有些问题很简单，有些则比较深奥；还有一些问题只是因为它们有趣才出现于书中。

本书涵盖C++的方方面面。我的意思并不是说它触及了C++的每一个细节（那将需要多得多的篇幅），而是指它从C++语言和库特性的广阔的原料中选取素材，向你展示看似无关的特性是如何被综合使用的，从而形成针对常见问题的新颖的解决方案。本书还展示了那些看似无关的部分是如何相互关联的（即便有时你不希望它们之间有什么关联）以及如何处理这些关联。你将在本书中看到关于模板与名字空间、异常与继承、健壮的类设计与设计模式、泛型编程与宏的使用魔法等内容。它们并非以随意花边新闻的形式出现，而是以具有内在联系的条款的形式，向你展示现代C++中所有这些部分之间的相互关系。

何谓“more”

《More Exceptional C++》从《Exceptional C++》停步的地方继续前行。本书继承了第一本书的传统：它以短小精悍的条款为组织形式，并将这些条款分组为主题明确的章节，以介绍新知识。读过第一本书的读者会发现一些熟悉的章节和主题，不过现在包含了新内容，例如异常安全、泛型编程以及内存管理技术等。这两本书在结构和主题而非内容上有重叠之处。

《More Exceptional C++》还有何不同之处呢？本书更加强调泛型编程和有效地使用C++标准库，包括对诸如特征萃取（traits）和判断式（predicates）这样的重要技术的讨论。一些条款对使用标准容器和算法时的注意事项提供了深入观察，其中许多注意事项我尚未在别处看到有所提及。一个新章节和两个附录集中于单线程和多线程环境下的优化，对于编写产品级代码的开发公司而言，这些议题现在比以往任何时候都

具有更实际的重要意义。

书中的大多数条款最初出现于因特网和杂志专栏上，尤其出自“Guru of the Week”[GotW]的议题31～62，以及我为《C/C++ Users Journal》、《Dr. Dobb's Journal》、《C++ Report》（已停刊）以及其他出版物撰写的专栏和文章。与最初的版本相比，本书中的材料经过大幅修订、扩充、校正和更新，因而本书（连同www.gotw.ca上不可或缺的勘误表）应被视作那些原始材料的最新权威版。

我假定你已经知道的

我期望你已经掌握了C++基础知识，如果你还没有，可以从一本介绍性和概览性的C++好书开始学习。像Bjarne Stroustrup的《The C++ Programming Language》（第3版）[Stroustrup00]或Stan Lippman和Josée Lajoie合著的《C++ Primer》（第3版）[Lippman98]这样的经典著作都是不错的选择。接下来，务必选读一本编程风格指南，例如Scott Meyers的经典著作《Effective C++》系列[Meyers96][Meyers97]。我发现基于浏览器的CD版[Meyers99]方便且实用。

如何阅读本书

书中的每一个条款都以谜题或问题的形式呈现，并带有一个介绍性的头部，如下所示：

Item##. 谜题的标题 Difficulty: X

标题和难度等级提示你将要遭遇到什么。注意，难度等级只是我自己预期大多数人认为问题有多难的主观猜测，因此，你也许会发现对你而言一个难度等级为7的问题比某个难度等级为5的问题更简单。自从写作《Exceptional C++》以来，我就不断收到一些电子邮件，说“条款N比你说的要容易（或困难）！”对于同一个条款而言，不同的人认为“更容易”或“更困难”是很正常的。难度等级因人而异，任何条款的实际难度取决于你的知识和经验，它们对于其他人而言则可能更容易或更困难一些。话虽如此，大多数情况下你应该发现这些难度等级对你将要看到的内容（的难度）还是给出了合理的提示。

你也许打算从头至尾按顺序阅读本书，这很好，但你未必非得这样不可。你也许决定阅读某个特定章节中的所有条款，因为你对该章节的主题特别感兴趣，这同样很好。除了被我称为“小型系列”的那些被标以“Part 1”、“Part 2”等条款讨论的是相关的问题外，书中条款通常是相当独立的，因此你可以遵循条款之间的交叉引用以及对《Exceptional C++》的引用，自由地跳着读。我唯一要忠告的是，那些“小型系列”要成组按顺序阅读，除此之外，如何阅读全凭你的喜好。

名字空间、typename、引用及其他约定

我在书中给出了不少建议，但我不会给你这样的指导方针：叫你去做连我自己都没做过的事。这包括我在整本书中我自己的示例代码中做的事。我还随既有实践和现代风格的大流，即便有时这并无实质性的影响。

在这方面，有必要说一下名字空间。在示例代码中，如果你在一个例子中看到一个文件范围的using指令，而在几页或几个条款之后的另一个例子中看到函数范围的using指令，这并没有什么深层次的原因，只是说明在该特定的情形下，那种用法是合适的，并且带给我美学上的愉悦，如是而已。如欲了解名字空间的基本原理，可阅读条款40。在正文中，当我想强调正在讨论的是标准库设施时，我会使用以std::限定的标准库名字，一旦确立了这一点，我通常重新使用非资格限定名字。

谈及模板参数的声明，我不时碰到一些人说写class而不是typename是过时的做法，即便这二者之间并无功能上的差别，而且标准文档自身都到处使用class。纯粹出于对风格的考虑，并且强调本书是关于今天的现代C++的，我已经转而使用typename而不是class来声明模板参数了。唯一的例外是在条款33中的某一处，由于那是我直接从标准中摘来的，标准使用的是class，我也就懒得动它了。

除非我明确指出某段代码是一个“完整的程序”，否则它很可能不是。记住，代码示例通常只是代码片段或部分的程序，别指望它们能被独立编译。为了从这些代码片段构建出完整的程序，通常你需要补充一些显而易见的边角代码。

最后，关于URL有必要多说一句：在Web上，东西会动来动去。尤其是，我无法控制的一些内容会动来动去。这使得在印刷书籍上刊印随意的Web URL就变成了真正的痛苦：恐怕在该书下厂印刷之前那些URL就已经过时了，更不要说等它在你的书桌上躺上5年之后了。当我在本书中引用他人的文章或Web站点时，我是通过自己的Web站点（www.gotw.ca）上的URL做到这一点的——我自己的Web站点是我所能控制的，它只包含对实际Web网页的重定向链接。如果你发现印刷在本书中的一个链接不再有效，请写邮件告诉我，我将更新该链接，使其指向新的网页位置（如果我还能找到该网页的话），或者告诉你该网页已不复存在（如果我找不到的话）。不管怎么说，本书的URL将会保持为最新，尽管在这个因特网世界中印刷媒体的日子是如此难过。呜呼！

致谢

非常感谢丛书编辑Bjarne Stroustrup，感谢Debbie Lafferty、Tyrrell Albaugh、Chanda Leary-Coutu、Charles Leddy、Curt Johnson以及Addison-Wesley团队的其他成员，感谢他们对这个项目的协助和坚持。很难想象还有比他们更好的共事伙伴，他们的热情和协作精神使我对这本书的所有期望都得到了实现。

还有一群人值得感谢和赞扬，就是许多专家审稿人。他们慷慨地提供了富有洞察力的见解并对书稿中的纰漏提出犀利的批评。他们的努力使得你手中的这本书更完整、更可读、更有用。特别感谢（排名大致以我收到审阅意见的顺序）Scott Meyers、Jan Christiaan van Winkel、Steve Dewhurst、Dennis Mancl、Jim Hyslop、Steve Clamage、Kevlin Henney、Andrew Koenig、Patrick McKillen以及一些不知名的审稿人。书中残存的任何错误、疏忽和不谦虚的双关语，与他们无关，责任全在我自己。

最后，特别感谢我的家人和朋友，感谢你们在本书写作过程中以及其他时间一直陪伴在我身边。

Herb Sutter

2001年6月于多伦多

Foreword

How do you become an expert? The answer is the same in all the fields I've seen:

1. Learn the basics.
2. Study the same material again—but this time, concentrate on the details you didn't realize were important the first time around.

If you pick the right details and master them so thoroughly that you no longer have to think about them, you will be much closer to being an expert. However, until you've become an expert, how do you know which details to pick? You'll learn a lot faster, and enjoy it more, if someone who's already been there picks the right details for you.

For example, I once took a photo workshop given by a fine photographer named Fred Picker. He told us that the only two hard parts of photography were where to put the camera and when to press the button. He then spent most of the workshop teaching us technical details about exposure, processing, and printing—details we had to absorb completely before we could control our photographs well enough for it even to make sense for us to concentrate on the two "hard" parts.

A particularly entertaining way to learn about the details of C++ programming is to try to answer questions about C++ programs. For example:

- Do `f(a++);` and `f(a); ++a;` have the same effect?
- Can you use an iterator to change the contents of a `set`?
- Suppose you're using a `vector` named `v` that has grown to use an uncomfortable amount of memory. You'd like to clear the `vector` and return that memory to the system. Will calling `v.clear()` do the trick?

You have probably guessed that the answers to these seemingly obvious questions must be no—otherwise I wouldn't have asked them—but do you know why the answers are no? Are you sure?

This book answers these questions and many other thoughtfully chosen questions about seemingly ordinary programs. There aren't many other books like it—except, of

course, its predecessor, *Exceptional C++*. Most C++ books that claim to be "advanced" are either about specialized topics—which is fine if you want to master those particular topics, but not if you are trying to look more deeply into everyday programs—or they use the word "advanced" merely to attract readers.

Once you understand these questions and answers thoroughly, you will no longer have to think so much about the details when you program; you will be free to concentrate on the problems you are really trying to solve.

Andrew Koenig
June 2001

此为试读,需要完整PDF请访问: www.ertongbook.com

Preface

The Greek philosopher Socrates taught by asking his students questions—questions designed to guide them and help them draw conclusions from what they already knew, and to show them how the things they were learning related to each other and to their existing knowledge. This method has become so famous that we now call it the "Socratic method." From our point of view as students, Socrates' approach involves us, makes us think, and helps us relate and apply what we already know to new information.

This book takes a page from Socrates, as did its predecessor, *Exceptional C++* [Sutter00]. It assumes you're involved in some aspect of writing production C++ software today, and uses a question-answer format to teach you how to make effective use of standard C++ and its standard library with a particular focus on sound software engineering in modern C++. Many of the problems are drawn directly from experiences I and others have encountered while working with production C++ code. The goal of the questions is to help you draw conclusions from things you already know as well as things you've just learned, and to show how they interrelate. The puzzles will show how to reason about C++ design and programming issues—some of them common issues, some not so common; some of them plain issues, some more esoteric; and a couple because, well, just because they're fun.

This book is about all aspects of C++. I don't mean to say that it touches on every detail of C++—that would require many more pages—but rather that it draws from the wide palette of the C++ language and library features to show how apparently unrelated items can be used together to synthesize novel solutions to common problems. It also shows how apparently unrelated parts of the palette interrelate on their own, even when you don't want them to, and what to do about it. You will find material here about templates and namespaces, exceptions and inheritance, solid class design and design patterns, generic programming and macro magic—and not just as randomized tidbits, but as cohesive Items showing the interrelationships among all of these parts of modern C++.

What's "More?"

More Exceptional C++ continues where *Exceptional C++* left off. This book follows in the tradition of the first: It delivers new material, organized in bite-sized Items and grouped into themed sections. Readers of the first book will find some familiar section themes, now including new material, such as exception safety, generic programming, and memory management techniques. The two books overlap in structure and theme, not in content.

Where else does *More Exceptional C++* differ? This book has a much stronger emphasis on generic programming and on using the C++ standard library effectively, including coverage of important techniques such as traits and predicates. Several Items provide in-depth looks at considerations to keep in mind when using the standard containers and algorithms; many of these considerations I've not seen covered elsewhere. There's a new section and two appendixes that focus on optimization in single- and multithreaded environments—issues that are now more than ever of practical consequence for development shops writing production code.

Versions of most Items originally appeared in Internet and magazine columns, particularly as *Guru of the Week* [GotW] issues #31 to 62, and as print columns and articles I've written for *C/C++ Users Journal*, *Dr. Dobb's Journal*, the former *C++ Report*, and other publications. The material in this book has been significantly revised, expanded, corrected, and updated since those initial versions, and this book (along with its *de rigueur* errata list available at *www.gotw.ca*) should be treated as the current and authoritative version of that original material.

What I Assume You Know

I expect that you already know the basics of C++. If you don't, start with a good C++ introduction and overview. Good choices are a classic tome like Bjarne Stroustrup's *The C++ Programming Language* [Stroustrup00], or Stan Lippman and Josée Lajoie's *C++ Primer, Third Edition* [Lippman98]. Next, be sure to pick up a style guide such as Scott Meyers' classic *Effective C++* books [Meyers96] [Meyers97]. I find the browser-based CD version [Meyers99] convenient and useful.

How to Read This Book

Each Item in this book is presented as a puzzle or problem, with an introductory header that resembles the following:

ITEM #: THE TOPIC OF THIS PUZZLE **DIFFICULTY: X**

The topic tag and difficulty rating gives you a hint of what you're in for. Note that the difficulty rating is my subjective guess at how difficult I expect most people will find each problem, so you may well find that a "7" problem is easier for you than some "5" problem. Since writing *Exceptional C++*, I've regularly received e-mail saying that "Item #N is easier (or harder) than that!" It's common for different people to vote "easier!" and "harder!" for the same Item. Ratings are personal; any Item's actual difficulty for you really depends on your knowledge and experience and could be easier or harder for someone else. In most cases, though, you should find the rating to be a good rule-of-thumb guide to what to expect.

You might choose to read the whole book front to back; that's great, but you don't have to. You might decide to read all the Items in a section together because you're particularly interested in that section's topic; that's cool, too. Except where there are what I call a "miniseries" of related problems which you'll see designated as "Part 1," "Part 2," and so on, the Items are pretty independent, and you should feel free to jump around, following the many cross-references among the Items in the book, as well as some references to *Exceptional C++*. The only guidance I'll offer is that the miniseries are designed to be read consecutively as a group; other than that, the choice is yours.

Namespaces, Typename, References, and Other Conventions

I make quite a few recommendations in this book, and I won't give you guidelines that tell you to do something I don't already do myself. That includes what I do in my own example code throughout this book. I'll also bow to existing practice and modern style, even when it really makes no material difference.

On that note, a word about namespaces: In the code examples, if you see a using-directive at file scope in one example and at function scope in another example a few pages or Items later, there's no deeper reason than that's what felt right and aesthetically pleasing to me for that particular case; for the rationale, turn to Item 40. In the narrative text itself, I've chosen to qualify standard library names with `std::` when I want to emphasize that it's the standard facility I'm talking about. Once that's established, I'll generally switch back to using the unqualified name.

When it comes to declaring template parameters, I sometimes come across people who think that writing `class` instead of `typename` is old-fashioned, even though there's no functional difference between the two and the standard itself uses `class`

most everywhere. Purely for style, and to emphasize that this book is about today's modern C++, I've switched to using `typename` instead of `class` to declare template parameters. The only exception is one place in Item 33, where I quote directly from the standard; the standard says `class`, so I left it in there.

Unless I call something a "complete program," it's probably not. Remember that the code examples are usually just snippets or partial programs and aren't expected to compile in isolation. You'll usually have to provide some obvious scaffolding to make a complete program out of the snippet shown.

Finally, a word about URLs: On the Web, stuff moves. In particular, stuff I have no control over moves. That makes it a real pain to publish random Web URLs in a print book lest they become out of date before the book makes it to the printer's, never mind after it's been sitting on your desk for five years. When I reference other people's articles or Web sites in this book, I do it via a URL on my own Web site, *www.gotw.ca*, which I can control and which contains just a straight redirect to the real Web page. If you find that a link printed in this book no longer works, send me e-mail and tell me; I'll update that redirector to point to the new page's location (if I can find the page again) or to say that the page no longer exists (if I can't). Either way, this book's URLs will stay up-to-date despite the rigors of print media in an Internet world. Whew.

Acknowledgments

Many thanks to series editor Bjarne Stroustrup, and to Debbie Lafferty, Tyrrell Albaugh, Chanda Leary-Coutu, Charles Leddy, Curt Johnson, and the rest of the Addison-Wesley team for their assistance and persistence during this project. It's hard to imagine a better bunch of people to work with, and their enthusiasm and cooperation has helped make this book everything I'd hoped it would become.

One other group of people deserves thanks and credit, namely the many expert reviewers who generously offered their insightful comments and savage criticisms exactly where they were needed. Their efforts have made the text you hold in your hands that much more complete, more readable, and more useful than it would otherwise have been. Special thanks to (in the approximate order that I received their review comments) Scott Meyers, Jan Christiaan van Winkel, Steve Dewhurst, Dennis Mancl, Jim Hyslop, Steve Clamage, Kevlin Henney, Andrew Koenig, Patrick McKillen, as well as several anonymous reviewers. The remaining errors, omissions, and shameless puns are mine, not theirs.

Finally, thanks most of all to my family and friends for always being there, during this project and otherwise.

Herb Sutter
Toronto, June 2001

Contents

此为试读,需要完整PDF请访问: www.ertongbook.com