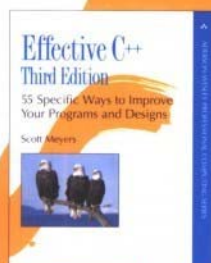


# Effective C++<sup>中文</sup>


Third Edition 第三版

55 Specific Ways to Improve  
Your Programs and Designs

改善程序与设计的55个具体做法



[美] Scott Meyers 著  
侯捷 译

 电子工业出版社  
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY  
<http://www.phei.com.cn>



ADDISON-WESLEY PROFESSIONAL COMPUTING SERIES

## Effective C++ 中文版, 第三版

“每一位 C++ 专业人士都需要这样一本《Effective C++》。对每一位想要认真以 C++ 从事开发工作的人而言, 这是一本绝对必须阅读的书。如果你不曾读过《Effective C++》却认为自己 C++ 无所不晓, 恐怕你得三思。”

— Steve Schirripa, Google 软件工程师

“C++ 和 C++ 社群已经在最近 15 年内成长了起来, 《Effective C++》第三版反映出这个事实。本书清晰而严谨的风格显示出 Scott 对于重要知识的深刻理解和特殊掌握能力。”

— Gerhard Kreuzer, Siemens AG 研发工程师

《Effective C++》前两个版本抓住了全世界无数程序员的目光。原因十分显明: Scott Meyers 极富实践意义的 C++ 研讨方式, 描述出专家用以产出干净、正确、高效代码的经验法则和行事法则——也就是他们几乎总是做或不做的某些事。

本书一共组织 55 个准则, 每一条准则描述一个编写出更好的 C++ 的方式。每一个条款的背后都有具体范例支撑。第三版有一半以上的篇幅是崭新内容, 包括讨论资源管理和模板 (templates) 运用的两个新章。为反映出现代设计考虑, 对第二版论题做了广泛的修订, 包括异常 (exceptions)、设计模式 (design patterns) 和多线程 (multithreading)。

《Effective C++》的重要特征包括:

- 高效的 classes、functions、templates 和 inheritance hierarchies (继承体系) 方面的专家级指导。
- 崭新的 “TR1” 标准程序库功能应用, 以及与既有标准程序库组件的比较。
- 洞察 C++ 和其他语言 (例如 Java、C#、C) 之间的不同。此举有助于那些来自其他语言阵营的开发人员消化吸收 C++ 式的各种解法。



## 作者简介

Scott Meyers 是全世界最知名的 C++ 软件开发专家之一。他是畅销书《Effective C++》系列 (Effective C++, More Effective C++, Effective STL) 的作者, 又是创新产品《Effective C++ CD》的设计者和作者, 也是 Addison-Wesley 的 “Effective Software Development Series” 顾问编辑, 以及《Software Development》杂志咨询板成员。他也为若干新公司的技术咨询板提供服务。Meyers 于 1993 年自 Brown 大学获得计算机博士学位。他的网址是 [www.aristeia.com](http://www.aristeia.com)。

译者简介: 侯捷是计算机技术书籍的作家、译者、书评人。著有《深入浅出MFC》、《多型与虚拟》、《STL源码剖析》、《无责任书评》三卷, 译有众多脍炙人口的高阶技术书籍, 包括 Meyers 所著的 “Effective C++” 系列。侯捷兼任教职于元智大学、同济大学、南京大学。他的个人网址是 <http://www.jihou.com> (中文繁体) 和 <http://jihou.csdn.net> (中文简体)。

图书分类: 程序语言

ISBN 7-121-02909-X



9 787121 029097 >

Addison-Wesley  
Pearson Education



网上订购: [www.dearbook.com.cn](http://www.dearbook.com.cn)  
第二书店·第一服务



责任编辑: 周 筠

本书贴有激光防伪标志, 凡没有防伪标志者, 属盗版图书。  
ISBN 7-121-02909-X 定价: 58.00元

# Effective C++ 中文版, 第三版

---

Effective C++, Third Edition

[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

有人说 C++ 程序员可以分成两类, 读过 Effective C++ 的和没读过的。世界顶级 C++ 大师 Scott Meyers 成名之作的第三版的确当得起这样的评价。当您读过这本书之后, 就获得了迅速提升自己 C++ 功力的一个契机。

在国际上, 本书所引起的反响, 波及整个计算技术出版领域, 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。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。

这本书不是读完一遍就可以束之高阁的快餐读物, 也不是用以解决手边问题的参考手册, 而是需要您去反复阅读体会的, C++ 是真正程序员的语言, 背后有着精深的思想与无与伦比的表达能力, 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风光, 做一个成功而快乐的 C++ 程序员。

Authorized translation from the English language edition, entitled Effective C++ : 55 Specific Ways to Improve Your Programs and Designs, 3<sup>rd</sup> edition, 0321334876 by Meyers, Scott, published by Pearson Education, Inc, publishing as Addison Wesley Professional, Copyright © 2005 Pearson Education, Inc.

All rights reserved. No part of this book may be reproduced or transmitted in any form or by any means, electronic or mechanical, including photocopying, recording or by any information storage retrieval system, without permission from Pearson Education, Inc.

CHINESE SIMPLIFIED language edition published by PEARSON EDUCATION ASIA LTD., and PUBLISHING HOUSE OF ELECTRONICS INDUSTRY Copyright © 2006.

本书简体中文版由电子工业出版社和 Pearson Education 培生教育出版亚洲有限公司合作出版。未经出版者预先书面许可, 不得以任何方式复制或抄袭本书的任何部分。

本书简体中文版贴有 Pearson Education 培生教育出版集团激光防伪标签, 无标签者不得销售。

版权贸易合同登记号: 图字: 01-2005-3583

## 图书在版编目 (CIP) 数据

Effective C++ 中文版, 第 3 版 / (美) 梅耶 (Meyers, S.) 著; 侯捷译. —北京: 电子工业出版社, 2006.7

书名原文: Effective C++, Third Edition

ISBN 7-121-02909-X

I. E... II. ①梅...②侯... III. C 语言—程序设计 IV. TP312

中国版本图书馆 CIP 数据核字 (2006) 第 081253 号

责任编辑: 周 筠

印 刷: 北京智力达印刷有限公司

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

经 销: 各地新华书店

开 本: 787×980 1/16 印张: 21 字数: 380 千字

印 次: 2006 年 7 月第 1 次印刷

定 价: 58.00 元

凡购买电子工业出版社的图书, 如有缺损问题, 请向购买书店调换。若书店售缺, 请与本社发行部联系。联系电话: (010) 68279077。质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

## Effective C++第三版赢得的赞誉

---

Scott Meyers的《Effective C++》第三版萃取了原本必须历经艰辛才能学到的编程经验。这本书是一份很棒的资源，我推荐给每一位专业C++ 程序员。

—— *Peter Dulimov, ME, Engineer, Ranges and Assessing Unit, NAVSYSCOM, Australia*

第三版仍然是“如何将 C++ 各部件以高效、高凝聚方式结合起来”的最佳书籍。声称自己是个 C++ 程序员之前，你一定得读过这本书。

—— *Eric Nagler, Consultant, Instructor, and author of Learning C++*

本书第一版被我归类为少数（真的非常少数）在我成长为一个专业软件开发人员的过程中有重大意义的书籍之一。它很实用又易阅读，却又装载着重要的忠告。

《Effective C++》第三版延续这项传统。C++ 是个威力十足的编程语言，如果C带给你足够绞死自己的绳索，C++ 就是间五金店，挤满了许多准备为你绑绳结的人。只要精通本书讨论的重点，便可明确增加高效运用C++ 的能力并减缓压力。

—— *Jack W. Reeves, Chief Executive Officer, Bleeding Edge Software Technologies*

每一位参与我的开发团队的新手，都有一份功课要做：读这本书。

—— *Michael Lanzetta, Senior Software Engineer*

九年前我读了《Effective C++》第一版，它立刻成为我最喜爱的一本 C++ 书籍。我认为第三版对于那些希望以C++ 进行高效编程的人仍然是必备读物。如果每一位 C++ 程序员着手写下他们的第一行C++ 专业代码之前都先读过这本书，我们的世界会变得更好一些。

—— *Danny Rabbani, Software Development Engineer*

当我还是个在第一线战场上努力搏斗的程序员，尝试怎么做比较好时，偶然机会遇上了Scott Meyers的《Effective C++》第一版。多美好的救星呀！我发现Meyers的忠告很实际、有用，并且有效，百分之百履行了标题上的承诺。第三版带来在严肃开发项目中使用C++ 的最新实用事物，并针对语言的最新发展和特性增加了新的篇章。我很高兴发现，从一本我原本以为自己已有很好体验的书籍的新版中，仍然学到一些有趣而新奇的东西。

—— *Michael Topic, Technical Program Manager*

对于想要安全并高效使用 C++，或打算从其它 OO 语言移转到 C++ 阵营的任何人而言，这一本来自著名 C++ 导师 Scott Meyers 的书籍，是最可靠的指引。本书以清晰、简洁、有娱乐效果、见解深刻的方式，表现出极具价值的信息。

—— *Siddhartha Karan Singh, Software Developer*

于一般性入门教科书之外，这应该是第二本任何 C++ 开发者应该阅读的书籍了。它超越了 C++ 语言“如何做”以及“是什么”的范畴，直指 C++ 的“为什么”。它帮助我对 C++ 的理解层次从语法晋升至编程哲学。

——Timothy Knox, *Software Developer*

这是一本 C++ 经典书籍的惊人更新版本。Meyers 在这一版本中涵盖了许多新领域，每一位认真的 C++ 程序员都应该拥有这一新版。

——Jeffrey Somers, *Game Programmer*

《Effective C++》第三版涵盖编写程序时该做的事，并很好地解释了为什么那些事情重要。把它视为编写 C++ 程序的最佳训练吧。

——Jeff Scherpelz, *Software Development Engineer*

当 C++ 拥抱改变，Scott Meyers 的《Effective C++》第三版也昂扬出发，对语言保持完美的密集跟踪。C++ 领域有许多优秀的导入性书籍，而“第二本书”应该站在它们的肩膀上，你手上这本就是。跟随 Scott 指出的方向，让自己也昂扬高飞吧！

——Leor Zolman, *C++ Trainer and Pundit, BD Software*

这是一本必须拥有的书籍，对 C++ 老手和新手都是。读过本书之后，它一定不会在你的书架上吃灰尘，因为你会持续地参考它、引用它。

——Sam Lee, *Software Developer*

阅读本书，一步一步地运用 55 个可轻松阅读并各自描述某项技术或某个告诫的条款，普通的 C++ 程序员也可以摇身一变成为专家级 C++ 程序员。

——Jeffrey D. Oldham, *Ph.D., Software Engineer, Google*

Scott Meyers 的《Effective C++》各个版本长期受到 C++ 编程新手和老手的需要。这本新版并入近十年来的 C++ 发展价值，是截至目前最高密度的书籍。作者不仅描述语言上的问题，也提出毫不模糊又容易奉行的忠告，用以避免陷阱并写出高效的 C++。我真希望每一位 C++ 程序员都能读过它。

——Philipp K. Janert, *Ph.D., Software Development Manager*

对那些使用 C++ 的时间长得足以被这一丰富语言内的潜伏圈套绊倒的开发人员而言，《Effective C++》的每个版本都是必须拥有的书籍。第三版大面积补充了新世代的语言和程序库特性，以及为运用那些特性而进化的编程风格。Scott 极具魅力的写作风格使其所整理的准则容易被消化吸收，协助你成为高效的 C++ 开发者。

——David Smallberg, *Instructor, DevelopMentor; Lecturer, Computer Science, UCLA*

《Effective C++》已针对 21 世纪的 C++ 实务做出全面更新，因此得以继续声称其为所有 C++ 从业人员的首选“第二本书”。

——Matthew Wilson, *Ph.D., author of Imperfect C++*

# **Effective C++**

## **Third Edition**

---

**55 Specific Ways to Improve Your Programs and Designs**

**Scott Meyers**

## Addison-Wesley Professional Computing Series

---

Brian W. Kernighan, Consulting Editor

Matthew H. Austern, *Generic Programming and the STL: Using and Extending the C++ Standard Template Library*

David R. Butenhof, *Programming with POSIX® Threads*

Brent Callaghan, *NFS Illustrated*

Tom Cargill, *C++ Programming Style*

William R. Cheswick/Steven M. Bellovin/Aviel D. Rubin, *Firewalls and Internet Security, Second Edition: Repelling the Wily Hacker*

David A. Curry, *UNIX® System Security: A Guide for Users and System Administrators*

Stephen C. Dewhurst, *C++ Gotchas: Avoiding Common Problems in Coding and Design*

Dan Farmer/Wietse Venema, *Forensic Discovery*

Erich Gamma/Richard Helm/Ralph Johnson/John Vlissides, *Design Patterns: Elements of Reusable Object-Oriented Software*

Erich Gamma/Richard Helm/Ralph Johnson/John Vlissides, *Design Patterns CD: Elements of Reusable Object-Oriented Software*

Peter Hagggar, *Practical Java™ Programming Language Guide*

David R. Hanson, *C Interfaces and Implementations: Techniques for Creating Reusable Software*

Mark Harrison/Michael McLennan, *Effective Tcl/Tk Programming: Writing Better Programs with Tcl and Tk*

Michi Henning/Steve Vinoski, *Advanced CORBA® Programming with C++*

Brian W. Kernighan/Rob Pike, *The Practice of Programming*

S. Keshav, *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network*

John Lakos, *Large-Scale C++ Software Design*

Scott Meyers, *Effective C++ CD: 85 Specific Ways to Improve Your Programs and Designs*

Scott Meyers, *Effective C++, Third Edition: 55 Specific Ways to Improve Your Programs and Designs*

Scott Meyers, *More Effective C++: 35 New Ways to Improve Your Programs and Designs*

Scott Meyers, *Effective STL: 50 Specific Ways to Improve Your Use of the Standard Template Library*

Robert B. Murray, *C++ Strategies and Tactics*

David R. Musser/Gillmer J. Derge/Atul Saini, *STL Tutorial and Reference Guide, Second Edition: C++ Programming with the Standard Template Library*

John K. Ousterhout, *Tcl and the Tk Toolkit*

Craig Partridge, *Gigabit Networking*

Radia Perlman, *Interconnections, Second Edition: Bridges, Routers, Switches, and Internetworking Protocols*

Stephen A. Rago, *UNIX® System V Network Programming*

Eric S. Raymond, *The Art of UNIX Programming*

Marc J. Rochkind, *Advanced UNIX Programming, Second Edition*

Curt Schimmel, *UNIX® Systems for Modern Architectures: Symmetric Multiprocessing and Caching for Kernel Programmers*

W. Richard Stevens, *TCP/IP Illustrated, Volume 1: The Protocols*

W. Richard Stevens, *TCP/IP Illustrated, Volume 3: TCP for Transactions, HTTP, NNTP, and the UNIX® Domain Protocols*

W. Richard Stevens/Bill Fenner/Andrew M. Rudoff, *UNIX Network Programming Volume 1, Third Edition: The Sockets Networking API*

W. Richard Stevens/Stephen A. Rago, *Advanced Programming in the UNIX® Environment, Second Edition*

W. Richard Stevens/Gary R. Wright, *TCP/IP Illustrated Volumes 1-3 Boxed Set*

John Viega/Gary McGraw, *Building Secure Software: How to Avoid Security Problems the Right Way*

Gary R. Wright/W. Richard Stevens, *TCP/IP Illustrated, Volume 2: The Implementation*

Ruixi Yuan/W. Timothy Strayer, *Virtual Private Networks: Technologies and Solutions*

Visit [www.awprofessional.com/series/professionalcomputing](http://www.awprofessional.com/series/professionalcomputing) for more information about these titles.



# Effective C++<sup>3/e</sup> 中文版

改善程序与设计的 55 个具体做法  
55 Specific Ways to Improve Your Programs and Designs

[美] Scott Meyers 著

侯捷 译

For Nancy,  
without whom nothing  
would be much worth doing

*Wisdom and beauty form a very rare combination.*

—— Petronius Arbiter  
Satyricon, XCIV

And in memory of Persephone.  
1995-2004



# 译序

按孙中山先生的说法，这个世界依聪明才智的先天高下得三种人：先知先觉得发明家，后知后觉得宣传家，不知不觉得实践家。三者之中发明家最少最稀珍，最具创造力。正是匠心独具的发明家创造了这个花花绿绿的计算机世界。

以文字、图书、授课形式来讲解、宣扬、引导技术的人，一般被视为宣传家而非发明家。然而，有一类最高等级的技术作家，不但能将精辟独到的见解诉诸文字，又能创造新的教学形式，引领风骚，对技术的影响和对产业的贡献不亚于技术或开发工具的创造者。这种人当之发明家亦无愧矣。

Scott Meyers 就是这一等级的技术作家！

自从 1991 年出版《Effective C++》之后，Meyers 声名大噪。1996 年的《More Effective C++》和 1997 年的《Effective C++》2/e 以及 2001 年的《Effective STL》让他更上高楼。Meyers 擅长探索编程语言的极限，穷尽其理，再以一支生花妙笔将复杂的探索过程和前因后果写成环环相扣故事性甚强的文字。他的幽默文风也让读者在高张力的技术学习过程中犹能享受“阅读的乐趣”——这是我对技术作家的最高礼赞。

以条款（items）传递专家经验，这种写作形式是否为 Meyers 首创我不确定，但的确是他造成了这种形式的计算机书籍写作风潮。影响所及，《Exceptional C++》、《More Exceptional C++》、《C++ Gotchas》、《C++ Coding Standards》、《Effective COM》、《Effective Java》、《Practical Java》纷纷在书名或形式上“向大师致敬”。

睽违 8 年之后《Effective C++》第三版面世了。我很开心继第二版再次受邀翻译。Meyers 在自序中对新版已有介绍，此处不待赘言。在此我适度修改第二版部分译序，援引于下，协助读者迅速认识本书定位。

C++ 是一个难学易用的语言!

C++ 的难学, 不仅在其广博的语法, 以及语法背后的语义, 以及语义背后的深层思维, 以及深层思维背后的对象模型; C++ 的难学还在于它提供了四种不同而又相辅相成的编程范型 (programming paradigms): procedural-based, object-based, object-oriented, generics.

世上没有白吃的午餐! 又要有效率, 又要弹性, 又要前瞻望远, 又要回溯相容, 又要治大国, 又要烹小鲜, 学习起来当然就不可能太简单。在庞大复杂的机制下, 万千使用者前仆后继的动力是: 一旦学成, 妙用无穷。

C++ 相关书籍车载斗量, 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*、*Thinking in C++*), 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++*), 独树一帜者有之 (*Design Patterns*, *Large Scale C++ Software Design*, *C++ FAQs*), 另辟蹊径者有之 (*Generic Programming and the STL*), 程序库大全有之 (*The C++ Standard Library*), 专家经验之累积亦有之 (*Effective C++*, *More Effective C++*)。这其中“专家经验之累积”对已具 C++ 相当基础的程序员有着立竿见影的帮助, 其特色是轻薄短小, 高密度纳入作者浸淫 C++/OOP 多年的广泛经验。它们不但开展读者的视野, 也为读者提供各种 C++/OOP 常见问题的解决模型。某些主题虽然在百科型 C++ 语言书中也可能提过, 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、最佳解法, 以及其他可能的牵扯。这些都是经验的累积和心血的结晶, 十分珍贵。

《Effective C++》就是这样一本轻薄短小高密度的“专家经验累积”。

本中译版与英文版页页对译, 保留索引, 偶尔加上小量译注; 愿能提供您一个愉快的学习。千里之行始于足下, 祝愿您从声名崇隆的本书展开一段新里程。同时, 我也向您推荐本书之兄弟《More Effective C++》, 那是 Meyers 的另一本同样盛名远播的书籍。

侯捷 2006/02/15 于台湾新竹

[jjhou@jjhou.com](mailto:jjhou@jjhou.com)

<http://www.jjhou.com> (繁体) <http://jjhou.csdn.net> (简体)

Effective C++中文版, 第三版

## 英中简繁术语对照

这里列出本书出现之编程术语的英中对照。本中文版在海峡两岸同步发行,因此我也列出本书简繁两版的术语对照,方便某些读者从中一窥两岸计算机用语。

表中带有 \* 者表示本书对该词条大多直接采用英文术语。中英术语的选择系由以下众多考虑中取其平衡:

- 业界和学界习惯。即便是学生读者,终也要离开学校进入职场;熟悉业界和学界的习惯用语(许多为英文),避免二次转换,很有必要。
- 这是一本中文版,需顾及中文阅读的感觉和顺畅性。过多保留英文术语会造成版面的破碎与杂乱!然若适度保留英文术语,可避免某些望之不像术语的中文出现于字里行间造成阅读的困扰和停顿,有助于流畅的思考和留下深刻印象。
- 凡涉及 C++ 语言关键字之相关术语皆保留。例如 `class`, `struct`, `template`, `public`, `private`, `protected`, `static`, `inline`, `const`, `namespace` ……
- 以上术语可能衍生复合术语,例如与 `class` 相关的复合术语有 `base class`, `derived class`, `super class`, `subclass`, `class template` …… 此类复合术语如果不长,尽皆保留原文;若太长则视情况另作处理(也许中英并陈,也许赋予特殊字体)。
- 凡计算机科学所称之数据结构名称,尽皆保留。例如 `stack`, `queue`, `tree`, `hashtable`, `map`, `set`, `deque`, `list`, `vector`, `array` …… 偶尔将 `array` 译为数组。
- 某些流通但不被我认为足够理想之中译词,保留原文不译。例如 `reference`。
- 某些英文术语被我刻意以特殊字体表现并保留,例如 *pass by reference*、*pass by value*、*copy* 构造函数、*assignment* 操作符、*placement new*。
- 少量术语为顾及词性平衡,时而采用中文(如指针、类型)时而采用英文(如 `pointer`、`type`)。
- 索引之于科技书籍非常重要。本书与英文版页页对译,因此原封不动保留所有英文索引。

过去以来我一直不甚满意 `object` 和 `type` 两个术语的中译词:“对象”和“类型”,认为它们缺乏术语突出性(前者正确性甚至有待商榷),却又频繁出现影响阅读,因此常在我的著作或译作中保留其英文词或偶尔采用繁体版术语:“物件”和“型别”。但现在我想,既然大家已经很习惯这两个中文术语,也许我只是杞人忧天。因此本书采用大陆读者普遍习惯的译法。不过我仍要提醒您,“`object`”在 Object Oriented 技术中的真正意义是“物体、物件”而非“对象、目标”。

以下带有 \* 者表示本书对该词条采英文词，不译为中文

英文术语	简体版译词	繁体版译词
abstract	抽象的	抽象的
abstraction	抽象性、抽象件	抽象性、抽象件
access	访问	存取、取用
access level	访问级别	存取級別
access function	访问函数	存取函式
adapter	适配器	配接器
address	地址	地址
address-of operator	取地址操作符	取址運算子
aggregation	聚合	聚合
algorithm	算法	演算法
allocate	分配	配置
allocator	分配器	配置器
application	应用程序	應用程式
architecture	体系结构	體系結構
argument	实参	引數
*array	数组	陣列
arrow operator	箭头操作符	箭頭運算子
assembly language	汇编语言	組合語言
*assert(-ion)	断言	斷言
assign(-ment)	赋值	賦值
assignment operator	赋值操作符	賦值運算子
*base class	基类	基礎類別
*base type	基类型	基礎型別
binary search	二分查找	二分搜尋
*binary tree	二叉树	二元樹
binary operator	二元操作符	二元運算子
binding	绑定	綁定、繫結
*bit	位	位元
*bitwise	(以 bit 为单元逐一 ……)	
block	区块	區塊
boolean	布尔值	布林值
breakpoint	断点	中斷點
build	建置	建置
build-in	内置	內建
bus	总线	匯流排
*byte	字节	位元組
cache	高速缓存(区)	快取(區)
call	调用	呼叫
callback	回调	回呼
call operator	call 操作符	call 運算子

英文术语	简体版译词	繁体版译词
character	字符	字元
*child class	子类	子類別
*class	类	類別
*class template	类模板	類別模板
client	客户	客戶
code	代码	程式碼
compatible	兼容	相容
compile time	编译期	編譯期
compiler	编译器	編譯器
component	组件	組件
composition	复合	複合
concrete	具象的	具象的
concurrent	并发	並行
configuration	配置	組態
connection	连接	連接, 連線
constraint	约束 (条件)	約束 (條件)
construct	构件	構件
container	容器	容器
*const	(C++ 关键字, 代表 constant)	
constant	常量	常數
constructor	构造函数	建構式
*copy (动词)	拷贝	拷貝、複製
copy (名词)	复件、副本	複件、副本
create	创建	產生、建立、生成
custom	定制	訂制、自定
data	数据	資料
database	数据库	資料庫
data member	成员变量	成員變數
data structure	数据结构	資料結構
debug	调试	除錯
debugger	调试器	除錯器
declaration	声明式	宣告式
default	缺省	預設
definition	定义式	定義式
delegate	委托	委託
dereference	提领 (解参考)	提領
*derived class	派生类	衍生類別
design pattern	设计模式	設計範式
destroy	销毁	銷毀
destructor	析构函数	解構式
directive	指示符	指令
document	文档	文件
dynamic binding	动态绑定	動態綁定



英文术语	简体版译词	繁体版译词
entity	物体	物體
encapsulation	封装	封裝
*enum(-eration)	枚举	列舉
equality	相等	相等
equivalence	等价	等價
evaluate	核定、核算	核定、核算
exception	异常	異常
explicit	显式	顯式、明白的
expression	表达式	算式
file	文件	檔案
framework	框架	框架
full specialization	全特化	全特化
function	函数	函式
function object	函数对象	函式物件
*function template	函数模板	函式模板
generic	泛型、泛化、一般化	泛型、泛化、一般化
*getter (相对于 setter)	取值函数	取值函式
*global	全局的	全域的
*handle	句柄	識別號、權柄
*handler	处理函数	處理函式
*hash table	哈希表、散列表	雜湊表
header (file)	头文件	表頭檔
*heap	堆	堆積
hierarchy	继承体系 (层次结构)	繼承體系 (階層體系)
identifier	标识符	識別字、識別符號
implement(-ation)	实现	實作
implicit	隐喻的、暗自的、隐式	隱喻的、暗自的、隱式
information	信息	資訊
inheritance	继承	繼承
*inline	内联	行內
initialization list	初值列	初值列
initialize	初始化	初始化
instance	实体	實體
instantiate	具现化、实体化	具現化、實體化
interface	接口	介面
Internet	互联网	網際網路
interpreter	解释器	直譯器
invariants	恒常性	恒常性
invoke	调用	喚起
iterator	迭代器	迭代器
library	程序库	程式庫
linker	连接器	連結器
literal	字面常量	字面常數

英文术语	简体版译词	繁体版译词
* list	链表	串列
load	装载	載入
* local	局部的	區域的
lock	机锁	機鎖
loop	循环	迴圈
lvalue	左值	左值
macro	宏	巨集
member	成员	成員
member function	成员函数	成員函式
memory	内存	記憶體
memory leak	内存泄漏	記憶體洩漏
meta-	元-	超-
* meta-programming	元编程	超編程
modeling	塑模	模塑
module	模块	模組
modifier	修饰符	飾詞
multi-tasking	多任务	多工
* namespace	命名空间	命名空間
native	固有的	原生的
nested	嵌套	嵌套、巢狀
object	对象	物件
object based	基于对象的	植基於物件、以物件為基礎
object model	对象模型	物件模型
object oriented	面向对象	物件導向
operand	操作数	運算元
operating system	操作系统	作業系統
operator	操作符	運算子
overflow	溢出	上限溢位
overhead	额外开销	額外開銷
overload	重载	重載
override	覆写	覆寫
package	包	套件
parallel	并行	平行
parameter	参数、形参	參數
* parent class	父类	父類別
parse	解析	解析
partial specialization	偏特化	偏特化
* pass by reference	按址传递	傳址
* pass by value	按值传递	傳值
pattern	模式	範式
* placement delete	(某种特殊形式的 delete operator)	
* placement new	(某种特殊形式的 new operator)	
pointer	指针	指標

英文术语	简体版译词	繁体版译词
polymorphism	多态	多型
preprocessor	预处理器	前處理器
print	打印	列印
printer	打印机	印表機
process	进程	行程
program	程序	程式
programmer	程序员	程式員
programming	编程	編程
project	项目	專案
pseudo code	伪码	偽碼
quality	质量	品質
*queue	队列	佇列
raw	原始的、未经处理的	原始的、未經處理的
recursive	递归	遞迴
refer to	指涉、指称、指向	指涉、指稱、指向
*reference	引用	參考、引用
regular expression	正则表达式	正則算式
resolve	解析	決議
return	返回	回返、傳回
return type	返回类型	回返型別
return value	返回值	回返值
runtime	运行期	執行期
rvalue	右值	右值
save	存储	儲存
schedule	调度	排程
scheduler	调度器	排程器
scope	作用域	作用域
search	查找	搜尋
semantics	语义	語意
*setter (相对于 getter)	设值函数	設值函式
signature	签名 (式)	簽名 (式)
smart pointer	智能指针	智慧型指標
specialization	特化	特化
source	源码	源碼
*stack	栈	堆疊
standard library	标准程序库	標準程式庫
statement	语句	述句
*static	静态的	靜態的
string	字符串	字串
*subtype	子类型	子型別
support	支持	支援
*template	模板	範本
temporary object	临时对象	暫時物件

英文术语	简体版译词	繁体版译词
text	文本	文字
thread	线程	緒程
thread safe	多线程安全	多緒安全
throw	抛、掷	拋、擲
token	语汇单元	語彙單元
type	类型	型別
unary function	单参函数	一元函式
underflow	下溢	下限溢位
unqualified	未经资格修饰	未經資格修飾
user	用户	用戶
user interface	用户界面	用戶介面
*value	值、数值	值、數值
variable	变量	變數
*vector	矢量	向量
virtual function	virtual 函数	virtual 函式
virtual machine	虚拟机	虛擬機器

# 序言

## Preface

1991 年我写下《*Effective C++*》第一版。1997 年撰写第二版时我更新了许多重要内容，但为了不让熟悉第一版的读者感到困惑，我竭尽所能保留原始结构：原先 50 个条款中的 48 个标题基本没变。如果把书籍视为一栋房屋，第二版只是更换地毯灯饰，重新粉刷一遍而已。

到了第三版，修缮工作进一步深入壁骨墙筋（好几次我甚至希望能够翻新地基）。1991 年起 C++ 世界经历了巨大变革，而本书目标——在一本小而有趣的书中确认最重要的一些 C++ 编程准则——却已不再能够由 15 年前建立的那些条款体现出来。“C++ 程序员拥有 C 背景”这句话在 1991 年是个合理假设，如今 C++ 程序员却很可能来自 Java 或 C# 阵营。继承（inheritance）和面向对象编程（object-oriented programming）在 1991 年对大多数程序员都很新鲜，如今程序员已经建立良好概念，异常（exceptions）、模板（templates）和泛型编程（generic programming）才是需要更多引导的领域。1991 年没人听过所谓设计模式（design patterns），如今少了它很难讨论软件系统。1991 年 C++ 正式标准才刚要上路，如今 C++ 标准规范已经 8 岁，新版规范蓄势待发。

为了对付这些改变，我把所有条款抹得一干二净，然后问自己“2005 年什么是对 C++ 程序员最重要的忠告？”答案便是第三版中的这些条款。本书有两个新章，一个是资源管理（resource management），一个是模板编程（programming with templates）。实际上 template（模板）这东西遍布全书，因为它们几乎影响了 C++ 的每个角落。本书新素材还包括在 exceptions（异常）概念下编程、套用设计模式、

以及运用新的 TR1 程序库设施（TR1 于条款 54 描述）。本书也告诉大家单线程系统（single-threaded systems）中运作良好但可能不适用于多线程系统（multithreaded systems）的某些技术和做法。本书半数以上内容是新的。在此同时第二版大部分基础信息仍然很重要，所以我找出一个保留它们的办法：你可以在附录 B 找到第二、第三两版的条款对应表。

我努力让本书达到我所能达到的最佳状态，但这并不表示它已臻完美。如果你认为某些条款不适合作为一般性忠告，或你有更好的办法完成本书所谈的某件工作，或书中某些技术讨论不够清楚不够完全，甚或有所误导，请告诉我。如果你找出任何错误——技术上的、文法上的、排版印刷上的，不论哪一种——也请告诉我。我很乐意将第一位提出问题并吸引我注意的朋友加入下次印刷的致谢名单中。

即使本书条款个数扩充为 55，这一整组编程准则还谈不上完备。然而毕竟整理出优良准则——几乎任何时间适用于任何应用程序的准则——比想象中困难得多。如果你有其他编程准则的想法或建议，我将乐以与闻。

我手上维护本书第一刷以来的变化清单，其中包括错误修订、进一步说明和技术更新。这份清单放在网址为 <http://aristeia.com/BookErrata/ec++3e-errata.html> 的 "Effective C++ Errata" 网页上。如果你希望在这份清单更新时获得通知，请加入我的邮件列表。这份列表用来发布消息给可能对我的专业工作感兴趣的人士，详情请见 <http://aristeia.com/MailingList/>。

Scott Douglas Meyers  
<http://aristeia.com/>

Stafford, Oregon  
April 2005

# 致谢

## Acknowledge

《Effective C++》已经面世 15 年了，我开始学习 C++ 则是在书写此书的前 5 年。也就是说 "Effective C++项目" 已经发展两个年代了。此期间我得益于数百（数千？）人的深刻知识、对我的建议与修正，以及偶发的一些目瞪口呆的事绩。这些人帮助我更加完善《Effective C++》，我要对他们全体表示感谢。

我已经放弃追踪“在哪儿学到什么”的历史，但永远记得有个公众信息来源不断带给我帮助：Usenet C++ newsgroups，特别是 comp.lang.c++.moderated 和 comp.std.c++。本书许多——也许是大多数——条款得益于这些讨论群所突出的若干技术想法和后续调查与诊疗。

关于第三版新内容，Steve Dewhurst 和我一起讨论了最初的条款名单。条款 11 中关于“藉由 copy-and-swap 实现 operator=”的构想来自 Herb Sutter 在此主题的作品，像是《Exceptional C++》（Addison-Wesley, 2000）条款 13。RAII（见条款 13）源自 Bjarne Stroustrup 的《The C++ Programming Language》（Addison-Wesley, 2000）。条款 17 背后的想法来自 Boost shared\_ptr 网页上的“Best Practices”节区（[http://boost.org/libs/smart\\_ptr/shared\\_ptr.htm#BestPractices](http://boost.org/libs/smart_ptr/shared_ptr.htm#BestPractices)），又得到 Herb Sutter 的《More Exceptional C++》（Addison-Wesley, 2002）条款 21 的琢磨。条款 29 强烈受到 Herb Sutter 在此主题上的广泛作品的影响，像是《Exceptional C++》条款 8~19，《More Exceptional C++》条款 17~23，以及《Exceptional C++ Style》（Addison-Wesley, 2005）条款 11~13；David Abrahams 帮助我更好地了解三个异常安全性保证。条款 35 的 NVI 手法来自 Herb Sutter 写于《C/C++ Users Journal》2001 年 9 月份的 "Virtuality" 专栏。同一条款中的 **Template Method** 和 **Strategy** 设计模式来自《Design Patterns》（Addison-Wesley, 1995），作者是 Erich Gamma, Richard Helm, Ralph Johnson 和 John Vlissides。条款 37 所说的 NVI 使用手法，概念来自

Hendrik Schober。David Smallberg 给了我动机在条款 38 写出一个定制型 `set` 实现品。条款 39 提到 EBO 通常只在多重继承中才可用, 这个构想源自 David Vandevoorde 和 Nicolai M. Josuttis 合著的《C++ Templates》(Addison-Wesley, 2003)。条款 42 中我对 `typename` 的最初理解来自 Greg Comeau 主持的 "C++ and C FAQ" (<http://www.comeaucomputing.com/techtalk/#typename>) , Leor Zolman 则帮助我认知我的最初了解并不正确 (是我的错, 和 Greg 无关)。条款 46 的本质源自于 Dan Sak 的谈话, "Making New Friends"。条款 52 末尾的那个想法 "如果你声明一版 `operator new`, 你也应该声明其对应的 `delete` 伙伴" 源自 Herb Sutter 的《Exceptional C++ Style》条款 22。我从 David Abrahams 身上更多了解了 Boost 的检评过程 (条款 55 有一份摘要)。

上面所说关于我向谁或从某处学习某一技术, 并不必然表示谁或某处就是该技术的发明人或发表处。

我的笔记还告诉我, 我也使用了来自 Steve Clamage, Antoine Trux, Timothy Knox 和 Mike Kaelbling 的信息, 可惜这份笔记没有提到如何以及在哪儿学到什么。

第一版草稿由 Tom Cargill, Glenn Carroll, Tony Davis, Brian Kernighan, Jak Kirman, Doug Lea, Moises Lejter, Eugene Santos, Jr., John Shewchuk, John Stasko, Bjarne Stroustrup, Barbara Tilly 和 Nancy L. Urbano 共同检阅。我收到了一些改善建议并纳入后来刷次, 这些建议来自 Nancy L. Urbano, Chris Treichel, David Corbin, Paul Gibson, Steve Vinoski, Tom Cargill, Neil Rhodes, David Bern, Russ Williams, Robert Brazile, Doug Morgan, Uwe Steinmüller, Mark Somer, Doug Moore, David Smallberg, Seth Meltzer, Oleg Shteynbuk, David Papurt, Tony Hansen, Peter McCluskey, Stefan Kuhlins, David Braunegg, Paul Chisholm, Adam Zell, Clovis Tondo, Mike Kaelbling, Natraj Kini, Lars Nyman, Greg Lutz, Tim Johnson, John Lakos, Roger Scott, Scott Frohman, Alan Rooks, Robert Poor, Eric Nagler, Antoine Trux, Cade Roux, Chandrika Gokul, Randy Mangoba 和 Glenn Teitelbaum。

第二版草稿由以下人士共同检阅: Derek Bosch, Tim Johnson, Brian Kernighan, Junichi Kimura, Scott Lewandowski, Laura Michaels, David Smallberg, Clovis Tondo, Chris Van Wyk 和 Oleg Zablude。我收到来自以下人士的意见并因此对新刷有所帮助: Daniel Steinberg, Arunprasad Marathe, Doug Stapp, Robert Hall, Cheryl Ferguson, Gary Bartlett, Michael Tamm, Kendall Beaman, Eric Nagler, Max Hailperin, Joe



Gottman, Richard Weeks, Valentin Bonnard, Jun He, Tim King, Don Maier, Ted Hill, Mark Harrison, Michael Rubenstein, Mark Rodgers, David Goh, Brenton Cooper, Andy Thomas-Cramer, Antoine Trux, John Wait, Brian Sharon, Liam Fitzpatrick, Bernd Mohr, Gary Yee, John O'Hanley, Brady Patterson, Christopher Peterson, Feliks Kluzniak, Isi Dunietz, Christopher Creutz, Ian Cooper, Carl Harris, Mark Stickel, Clay Budin, Panayotis Matsinopoulos, David Smallberg, Herb Sutter, Pajo Misljencevic, Giulio Agostini, Fredrik Blomqvist, Jimmy Snyder, Byrial Jensen, Witold Kuzminski, Kazunobu Kuriyama, Michael Christensen, Jorge Yáñez Teruel, Mark Davis, Marty Rabinowitz, Ares Lagae 和 Alexander Medvedev.

第三版早期部分草稿由以下人士共同检阅: Brian Kernighan, Angelika Langer, Jesse Laeuchli, Roger E. Pedersen, Chris Van Wyk, Nicholas Stroustrup 和 Hendrik Schober. 完整草稿由以下人士共同检阅: Leor Zolman, Mike Tsao, Eric Nagler, Gene Gutnik, David Abrahams, Gerhard Kreuzer, Drosos Kourounis, Brian Kernighan, Andrew Kirmse, Balog Pal, Emily Jagdhar, Eugene Kalenkovich, Mike Roze, Enrico Carrara, Benjamin Berck, Jack Reeves, Steve Schirripa, Martin Fallenstedt, Timothy Knox, Yun Bai, Michael Lanzetta, Philipp Janert, Guido Bartolucci, Michael Topic, Jeff Scherpelz, Chris Nauroth, Nishant Mittal, Jeff Somers, Hal Moroff, Vincent Manis, Brandon Chang, Greg Li, Jim Meehan, Alan Geller, Siddhartha Singh, Sam Lee, Sasan Dashtinezhad, Alex Marin, Steve Cai, Thomas Fruchterman, Cory Hicks, David Smallberg, Gunavardhan Kakulapati, Danny Rabbani, Jake Cohen, Hendrik Schober, Paco Viciano, Glenn Kennedy, Jeffrey D. Oldham, Nicholas Stroustrup, Matthew Wilson, Andrei Alexandrescu, Tim Johnson, Leon Matthews, Peter Dulimov 和 Kevlin Henney. 某些个别条款的草稿由 Herb Sutter 和 Attila F. Feher 检阅。

检阅一份不够洗炼（而且可能尚未完成）的手稿是件吃力的工作，在时间压力之下进行只会使得它更困难。我要感谢这么多人愿意为我做这件事。

如果对讨论素材缺乏背景，而又期望捕捉手稿中的每一个问题，检阅工作将更加困难。令人惊讶的是还是有人选择成为文字编辑。Chrysta Meadowbrooke 是本书的文字编辑，她的周密工作揭露出许多逃过其他每一双眼睛的问题。

Leor Zolman 在正式检阅前先以多种编译器检验所有代码，在我校订手稿之后又做一次。如果书中仍然存在任何错误，全是我的责任。Karl Wieggers 和（特别是）

Tim Johnson 提供我快速而有帮助的反饋。

John Wait 是我的前两版编辑，有点傻傻地又签下这一份责任约。他的助理 Denise Mickelsen 熟练地处理我频繁的纠缠，带着愉快的微笑（至少我认为她是。呃，我从未见过她）。Julie Nahil 向来扮演救火队角色并因此成为我的产品经理。她以非凡的镇定指挥产品计划内的六周通宵工作。John Fuller（她的老板）和 Marty Rabinowitz（他的老板）也协助解决了产品发行量问题。Vanessa Moore 的正式工作是提供 FrameMaker 和 PDF 支持，但她也协助制作附录 B 的条目并格式化打印于封底里。Solveig Haugland 协助将索引格式化。Sandra Schroeder 和 Chuti Prasertsith 负责封面设计，Chuti 总是在每次我说“如果把这张相片加上那个颜色的线条会怎样？”时修订封面。Chanda Leary-Coutu 对市场营销举重若轻。

在我忙于手稿的数月之中，电视剧集 *Buffy the Vampire Slayer* 常常帮助我在每天工作结束后解压。带着极大的克制我才能够不让 Buffyspeak 的身影进入本书。Kathy Reed 于 1971 年教我写程序，我很高兴我们保持友谊至今。Donald French 雇用我和 Moises Lejter 于 1989 年建立起 C++ 培训教材（这项计划诱使我真正了解 C++），1991 年他又聘我在 Stratus Computer 体现它们。该班学生鼓励我写下最终成为本书第一版的东西。Don 也将我介绍给 John Wait，他同意出版它们。

我的妻子 Nancy L. Urbano 持续鼓励我写作，即使在我完成了七本书、一张 CD 改写版、一篇论文之后。她有令人难以置信的忍耐、自制与宽容。没有她我无法完成我所完成的任何事情。

自始至终，我们的狗儿 Persephone 是我们无可取代的同伴。令人悲伤的是，在这个项目的大部分时间里，她和我们之间的交往关系已经改为办公室内的一坛骨灰瓮。我们十分怀念她。