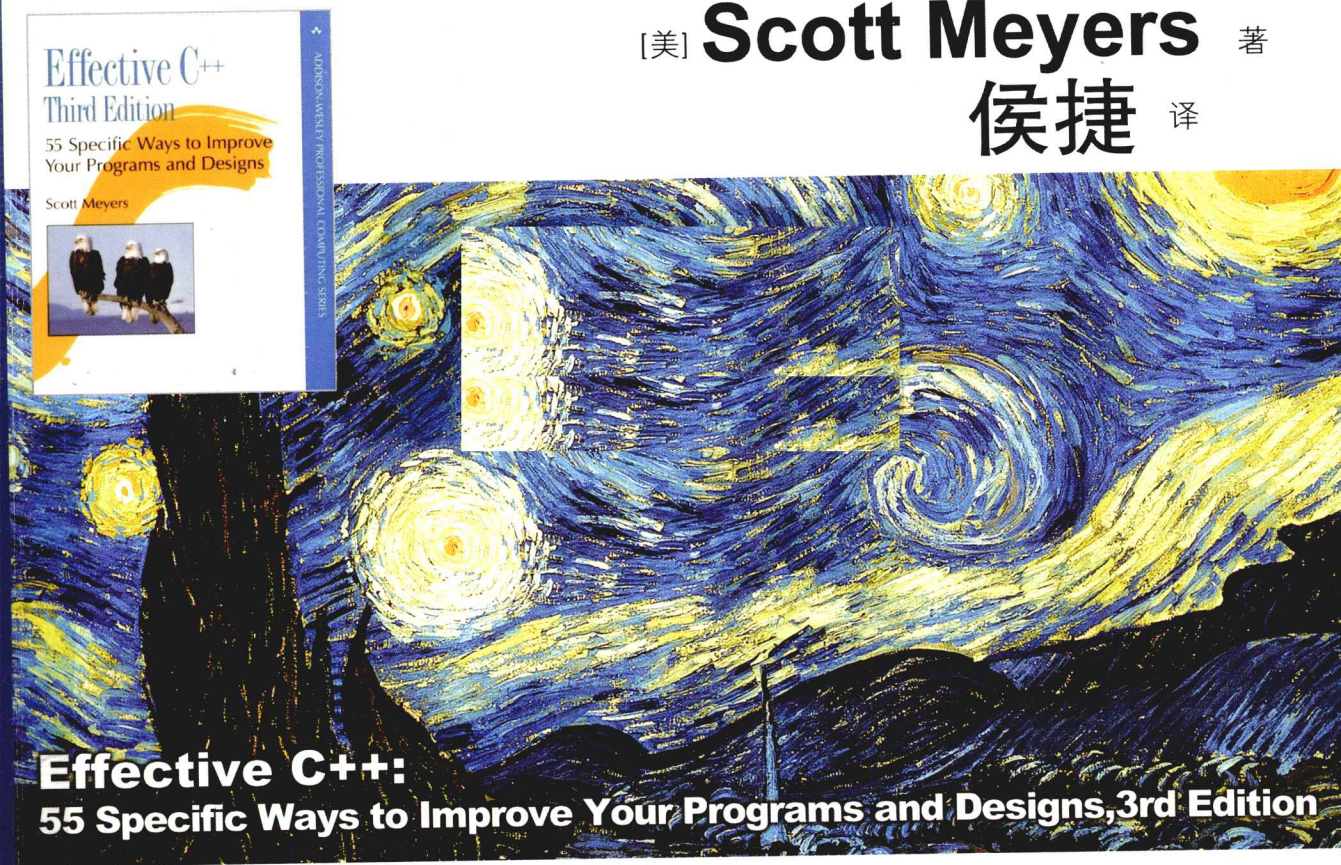
Broadview®
www.broadview.com.cn
PEARSON
传世经典书丛
Eternal Classics

Effective C++ 中文版

改善程序与设计的55个具体做法 (第三版)

[美] Scott Meyers 著
侯捷 译



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

Effective C++

改善程序与设计的55个具体做法（第三版）中文版

Effective C++ : 55 Specific Ways to Improve Your Programs and Designs

3rd Edition

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

内 容 简 介

有人说C++程序员可以分为两类，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，波及整个计算机技术的出版领域，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。

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++：改善程序与设计的55个具体做法：第3版 /（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：Effective C++ : 55 Specific Ways to Improve Your Programs and Designs, 3/e
ISBN 978-7-121-12332-0

Ⅰ. ①E… 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223862号

责任编辑：周 筠
文字编辑：许 艳
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：380千字
印 次：2011年1月第1次印刷
定 价：65.00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

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, Bleading 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++ 3/e

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

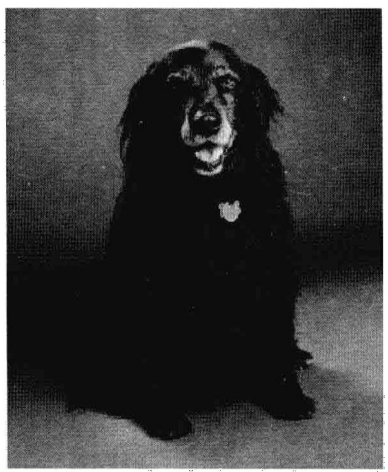
侯捷 译

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，又要有弹性，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
http://www.jjhou.com（繁体） http://jjhou.csdn.net（简体）

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，可避免某些望之不似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、以物件爲基礎
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